Pieter Johannes (600640)

Table of Contents

1. Applications and Containers (Docker)

We'll create three backend services and one frontend:

- book-catalog-service

- cart-service

- order-service

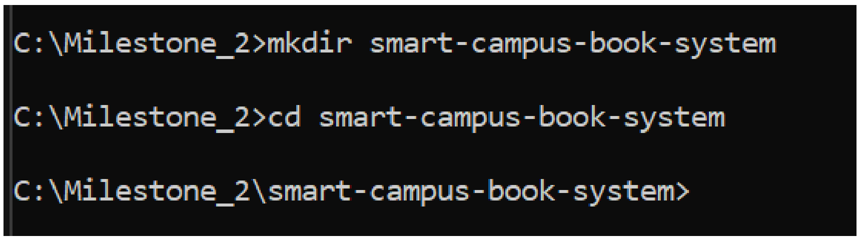
- web-ui "frontend"

We'll use Python and Dockerize each.

Step 1: Create Project Folder

mkdir smart-campus-book-system

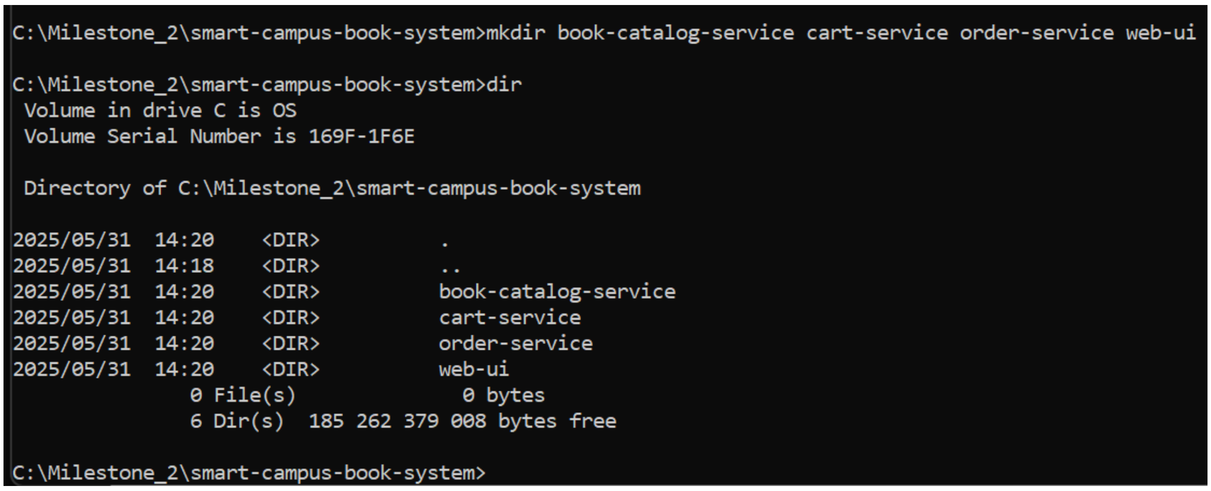
cd smart-campus-book-system



```
C:\Milestone_2>mkdir smart-campus-book-system

C:\Milestone_2>cd smart-campus-book-system

C:\Milestone_2\smart-campus-book-system>
```

Create a folder for each microservice:

mkdir book-catalog-service cart-service order-service web-ui



```
C:\Milestone_2\smart-campus-book-system>mkdir book-catalog-service cart-service order-service web-ui

C:\Milestone_2\smart-campus-book-system>dir
 Volume in drive C is OS
 Volume Serial Number is 169F-1F6E

 Directory of C:\Milestone_2\smart-campus-book-system

2025/05/31  14:20    <DIR>          .
2025/05/31  14:18    <DIR>          ..
2025/05/31  14:20    <DIR>          book-catalog-service
2025/05/31  14:20    <DIR>          cart-service
2025/05/31  14:20    <DIR>          order-service
2025/05/31  14:20    <DIR>          web-ui
               0 File(s)              0 bytes
               6 Dir(s)  185 262 379 008 bytes free

C:\Milestone_2\smart-campus-book-system>
```

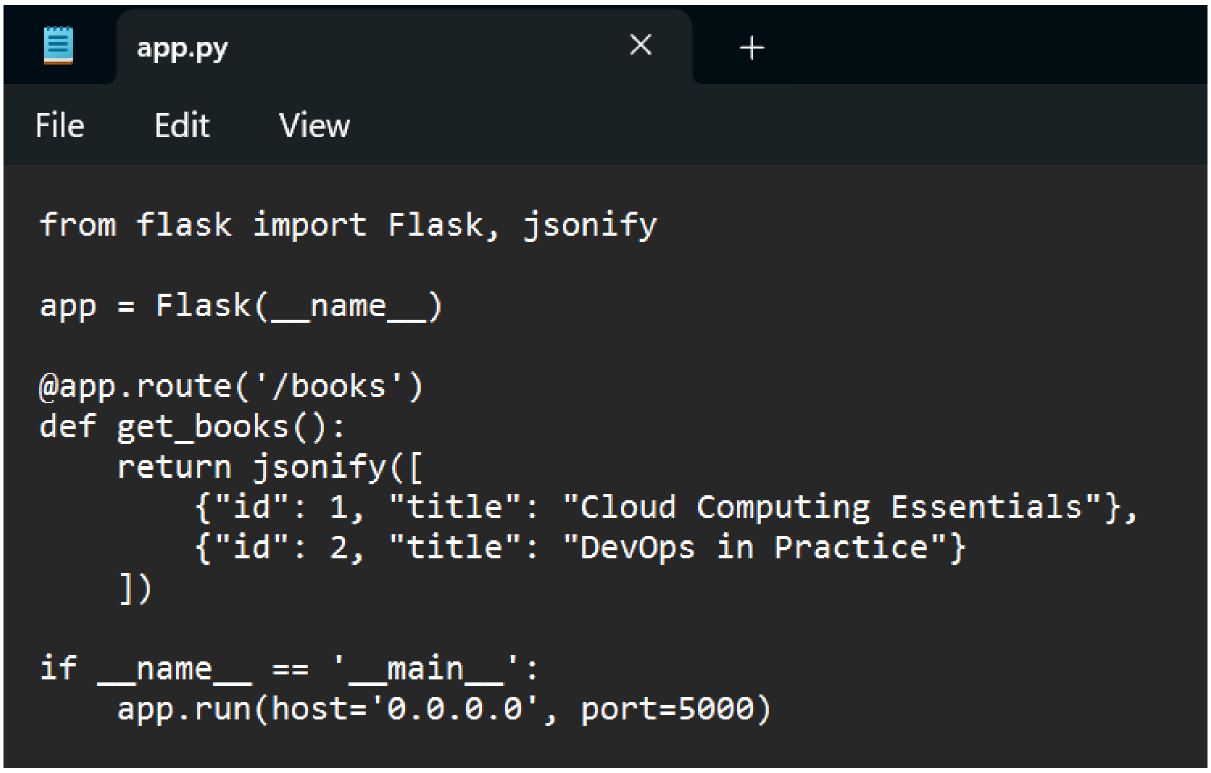
Step2: Build a Sample Flask App for One Service

Let’s creat a folder with the name "book-catalog-service".

creat a file name "app.py" inside the folder

the code:

```
from flask import Flask, jsonify


app = Flask(__name__)


@app.route('/books')
def get_books():
    return jsonify([
        {"id": 1, "title": "Cloud Computing Essentials"},
        {"id": 2, "title": "DevOps in Practice"}
    ])


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```



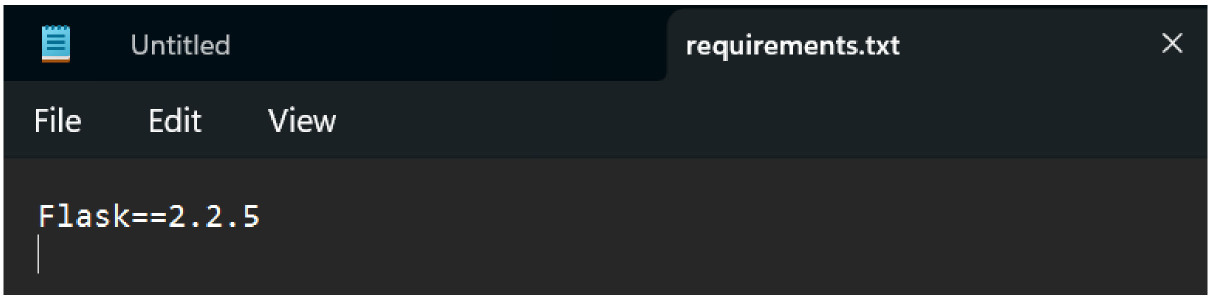
```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/books')
def get_books():
    return jsonify([
        {"id": 1, "title": "Cloud Computing Essentials"},
        {"id": 2, "title": "DevOps in Practice"}
    ])

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

----

creat a file name "requirements.txt"

the Code:

Flask==2.2.5



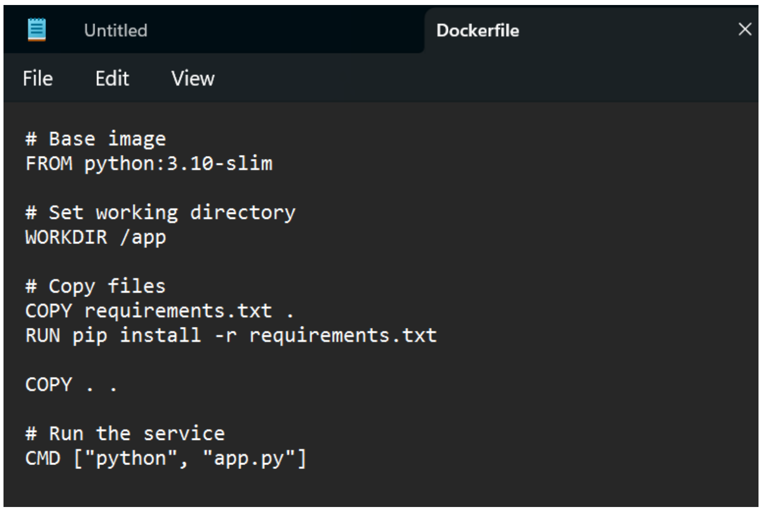
```
Flask==2.2.5
```

----

step3: Create a Dockerfile

the code:

```
# Base image
FROM python:3.10-slim

# Set working directory
WORKDIR /app

# Copy files
COPY requirements.txt .
RUN pip install -r requirements.txt

COPY . .

# Run the service
CMD ["python", "app.py"]
```



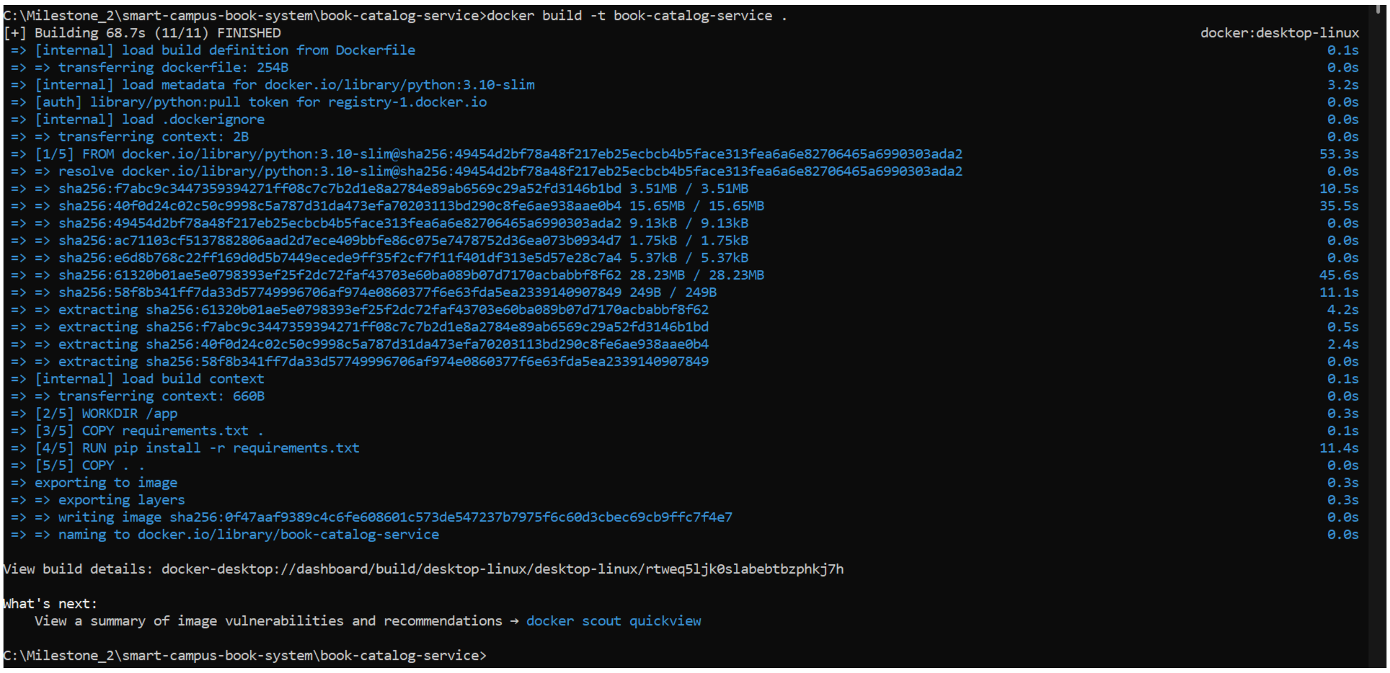
```
# Base image
FROM python:3.10-slim

# Set working directory
WORKDIR /app

# Copy files
COPY requirements.txt .
RUN pip install -r requirements.txt

COPY . .

# Run the service
CMD ["python", "app.py"]
```

Step4: Build and Run the Docker Container

Run docker engin => type: **Docker login**

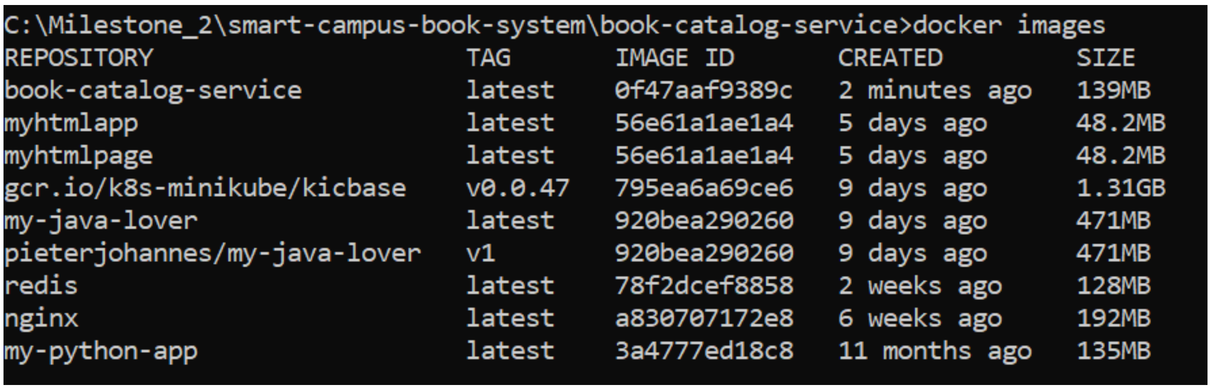
Make sure you’re inside the book-catalog-service folder:

docker build -t book-catalog-service .



```
C:\Milestone_2\smart-campus-book-system\book-catalog-service>docker build -t book-catalog-service .
[+] Building 68.7s (11/11) FINISHED                                                                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                                                 0.1s
 => => transferring dockerfile: 254B                                                                                                                                 0.0s
 => [internal] load metadata for docker.io/library/python:3.10-slim                                                                                                  3.2s
 => [auth] library/python:pull token for registry-1.docker.io                                                                                                        0.0s
 => [internal] load .dockerignore                                                                                                                                    0.0s
 => => transferring context: 2B                                                                                                                                      0.0s
 => [1/5] FROM docker.io/library/python:3.10-slim@sha256:49454d2bf78a48f217eb25ecbcb4b5face313fea6a6e82706465a6990303ada2                                           53.3s
 => => resolve docker.io/library/python:3.10-slim@sha256:49454d2bf78a48f217eb25ecbcb4b5face313fea6a6e82706465a6990303ada2                                            0.0s
 => => sha256:f7abc9c3447359394271ff08c7c7b2d1e8a2784e89ab6569c29a52fd3146b1bd 3.51MB / 3.51MB                                                                      10.5s
 => => sha256:40f0d24c02c50c9998c5a787d31da473efa70203113bd290c8fe6ae938aae0b4 15.65MB / 15.65MB                                                                    35.5s
 => => sha256:49454d2bf78a48f217eb25ecbcb4b5face313fea6a6e82706465a6990303ada2 9.13kB / 9.13kB                                                                       0.0s
 => => sha256:ac71103cf5137882806aad2d7ece409bbfe86c075e7478752d36ea073b0934d7 1.75kB / 1.75kB                                                                       0.0s
 => => sha256:e6d8b768c22ff169d0d5b7449ecede9ff35f2cf7f11f401df313e5d57e28c7a4 5.37kB / 5.37kB                                                                       0.0s
 => => sha256:61320b01ae5e0798393ef25f2dc72faf43703e60ba089b07d7170acbabbf8f62 28.23MB / 28.23MB                                                                    45.6s
 => => sha256:58f8b341ff7da33d57749996706af974e0860377f6e63fda5ea2339140907849 249B / 249B                                                                          11.1s
 => => extracting sha256:61320b01ae5e0798393ef25f2dc72faf43703e60ba089b07d7170acbabbf8f62                                                                            4.2s
 => => extracting sha256:f7abc9c3447359394271ff08c7c7b2d1e8a2784e89ab6569c29a52fd3146b1bd                                                                            0.5s
 => => extracting sha256:40f0d24c02c50c9998c5a787d31da473efa70203113bd290c8fe6ae938aae0b4                                                                            2.4s
 => => extracting sha256:58f8b341ff7da33d57749996706af974e0860377f6e63fda5ea2339140907849                                                                            0.0s
 => [internal] load build context                                                                                                                                    0.1s
 => => transferring context: 660B                                                                                                                                    0.0s
 => [2/5] WORKDIR /app                                                                                                                                               0.3s
 => [3/5] COPY requirements.txt .                                                                                                                                    0.1s
 => [4/5] RUN pip install -r requirements.txt                                                                                                                       11.4s
 => [5/5] COPY . .                                                                                                                                                   0.0s
 => exporting to image                                                                                                                                               0.3s
 => => exporting layers                                                                                                                                              0.3s
 => => writing image sha256:0f47aaf9389c4c6fe608601c573de547237b7975f6c60d3cbec69cb9ffc7f4e7                                                                         0.0s
 => => naming to docker.io/library/book-catalog-service                                                                                                              0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/rtweq5ljk0slabebtbzphkj7h

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\Milestone_2\smart-campus-book-system\book-catalog-service>
```

Type **docker images**

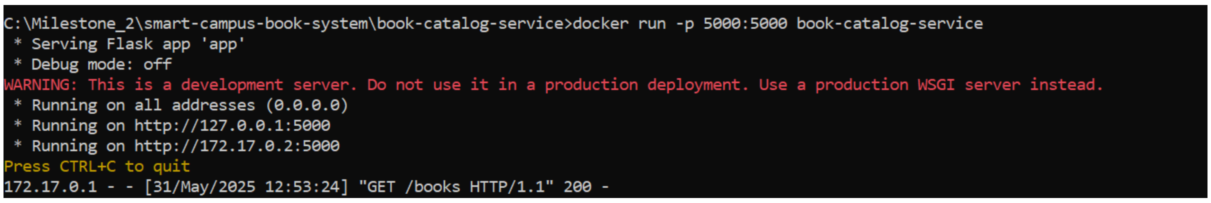


```
C:\Milestone_2\smart-campus-book-system\book-catalog-service>docker images
REPOSITORY                     TAG       IMAGE ID       CREATED          SIZE
book-catalog-service           latest    0f47aaf9389c   2 minutes ago    139MB
myhtmlapp                      latest    56e61a1ae1a4   5 days ago       48.2MB
myhtmlpage                     latest    56e61a1ae1a4   5 days ago       48.2MB
gcr.io/k8s-minikube/kicbase    v0.0.47   795ea6a69ce6   9 days ago       1.31GB
my-java-lover                  latest    920bea290260   9 days ago       471MB
pieterjohannes/my-java-lover   v1        920bea290260   9 days ago       471MB
redis                          latest    78f2dcef8858   2 weeks ago      128MB
nginx                          latest    a830707172e8   6 weeks ago      192MB
my-python-app                  latest    3a4777ed18c8   11 months ago    135MB
```

You will see book-catalog-service on the top

Type this and run the container:

docker run -p 5000:5000 book-catalog-service



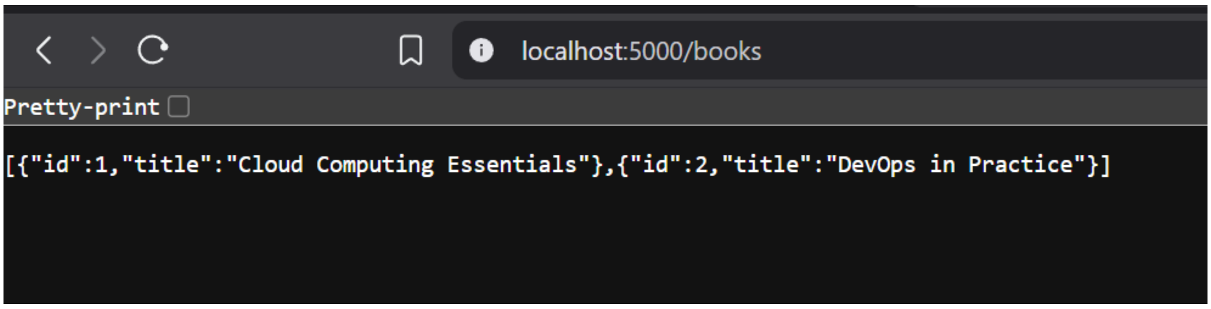
```
C:\Milestone_2\smart-campus-book-system\book-catalog-service>docker run -p 5000:5000 book-catalog-service
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (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
172.17.0.1 - - [31/May/2025 12:53:24] "GET /books HTTP/1.1" 200 -
```

- This means you're successfully hitting the /books endpoint and it's returning HTTP 200, it is successful response.

- You accessed it from the browser or a tool like Postman.

Type this into the browser you should see JSON data.

Visit http://localhost:5000/books in your browser



localhost:5000/books

Pretty-print

```
[{"id":1,"title":"Cloud Computing Essentials"},{"id":2,"title":"DevOps in Practice"}]
```

This is a JSON response from the "/books" endpoint. It shows a list of book objects, each with an id and a title.

This proves:

- Flask app is working inside Docker.

- "/books route" is accessible from your browser.

- JSON response is being served correctly.

- Nginx or any other reverse proxy is not needed yet because it's a development setup.

Repeat for Other Microservices

Each of these will follow the same structure:

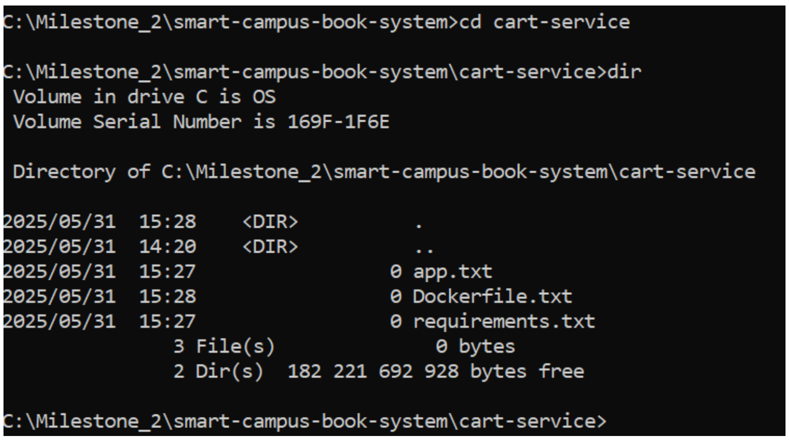
- cart-service: manages selected books
- order-service: handles checkout/orders

Create cart-service Microservice

Step 1: Folder Structure

go on cart-service

cd cart-service



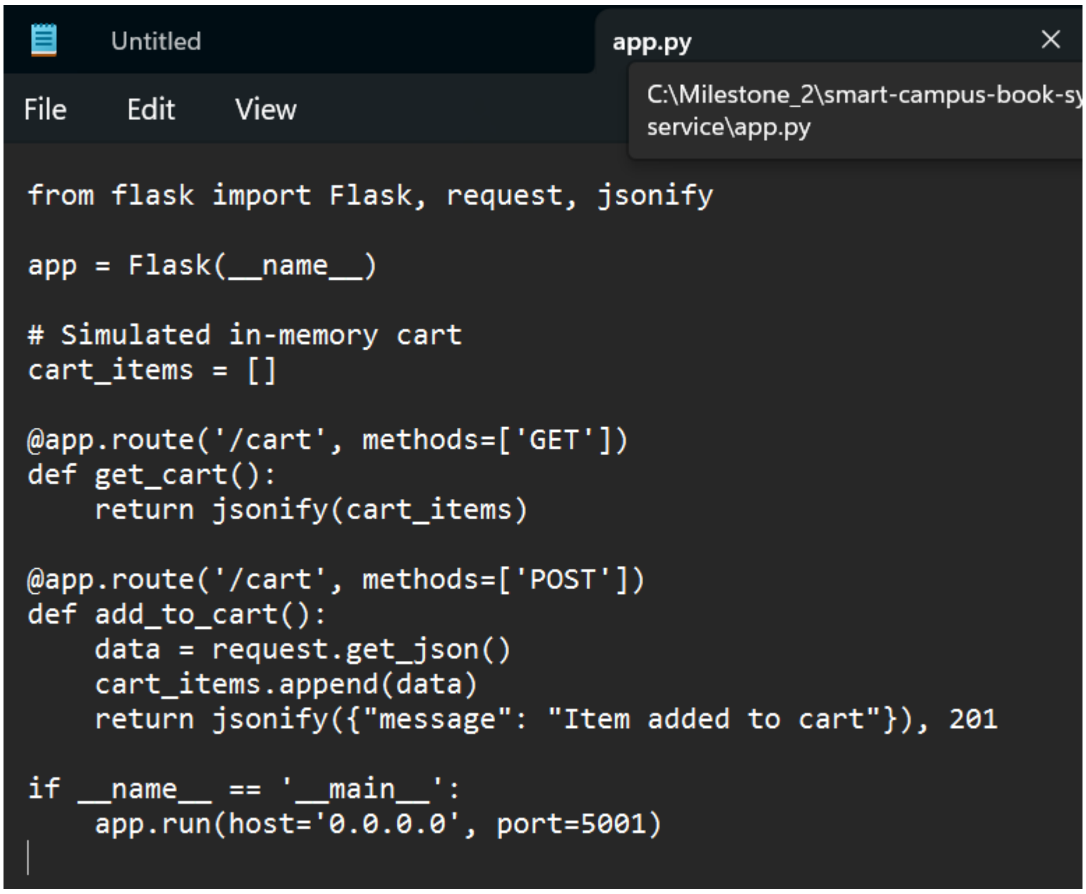
```
C:\Milestone_2\smart-campus-book-system>cd cart-service

C:\Milestone_2\smart-campus-book-system\cart-service>dir
 Volume in drive C is OS
 Volume Serial Number is 169F-1F6E

 Directory of C:\Milestone_2\smart-campus-book-system\cart-service

2025/05/31  15:28    <DIR>          .
2025/05/31  14:20    <DIR>          ..
2025/05/31  15:27                 0 app.txt
2025/05/31  15:28                 0 Dockerfile.txt
2025/05/31  15:27                 0 requirements.txt
               3 File(s)              0 bytes
               2 Dir(s)  182 221 692 928 bytes free

C:\Milestone_2\smart-campus-book-system\cart-service>
```

Step 2: app.py – Flask App Code

Create a file called "app.py" with this code:

```
from flask import Flask, request, jsonify

app = Flask(__name__)

# Simulated in-memory cart
cart_items = []

@app.route('/cart', methods=['GET'])
def get_cart():
  return jsonify(cart_items)
```

```
@app.route('/cart', methods=['POST'])

def add_to_cart():

    data = request.get_json()

    cart_items.append(data)

    return jsonify({"message": "Item added to cart"}), 201


if __name__ == '__main__':

    app.run(host='0.0.0.0', port=5001)
```



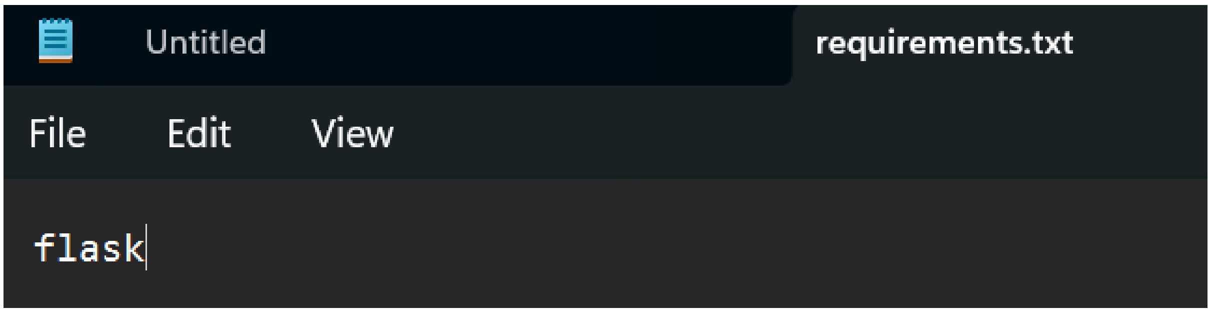
```
from flask import Flask, request, jsonify

app = Flask(__name__)

# Simulated in-memory cart
cart_items = []

@app.route('/cart', methods=['GET'])
def get_cart():
    return jsonify(cart_items)

@app.route('/cart', methods=['POST'])
def add_to_cart():
    data = request.get_json()
    cart_items.append(data)
    return jsonify({"message": "Item added to cart"}), 201

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001)
```

Step 3: requirements.txt

Create a file called requirements.txt:

the code:

```
flask
```

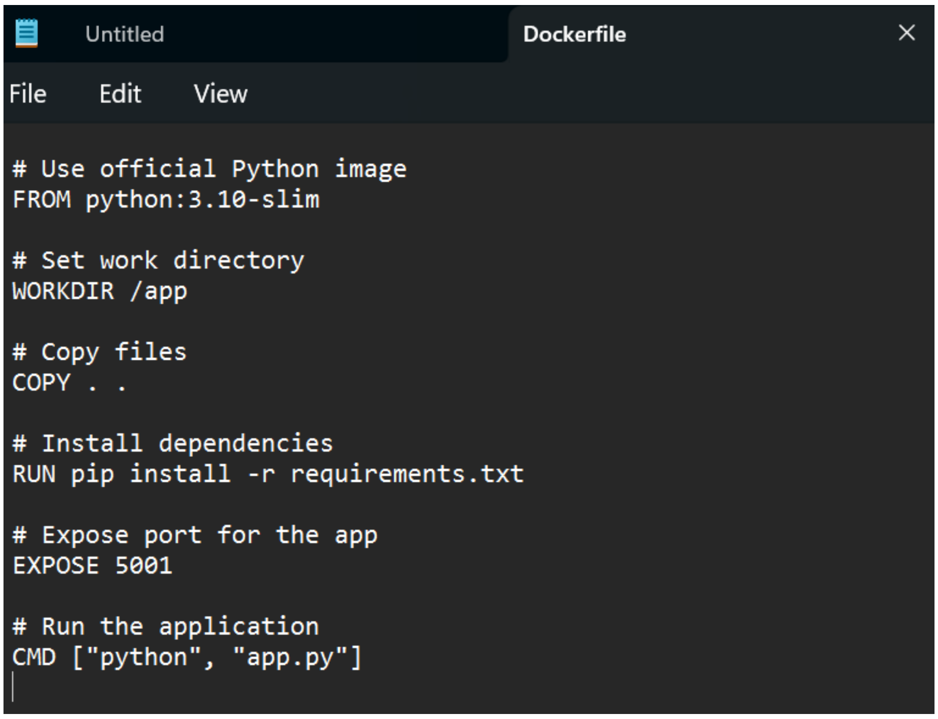


```
flask
```

Step 4: Dockerfile

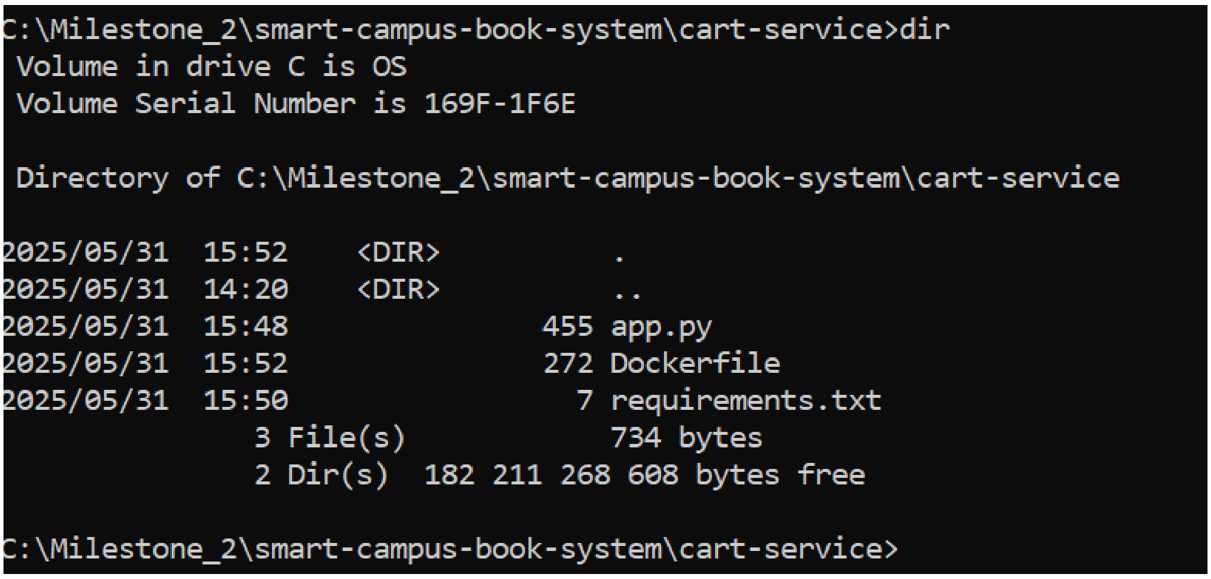
create a Dockerfile:

```
# Use official Python image
FROM python:3.10-slim

# Set work directory
WORKDIR /app

# Copy files
COPY . .

# Install dependencies
RUN pip install -r requirements.txt

# Expose port for the app
EXPOSE 5001

# Run the application
CMD ["python", "app.py"]
```



```
# Use official Python image
FROM python:3.10-slim

# Set work directory
WORKDIR /app

# Copy files
COPY . .

# Install dependencies
RUN pip install -r requirements.txt

# Expose port for the app
EXPOSE 5001

# Run the application
CMD ["python", "app.py"]
```

Type in dir in cmd:

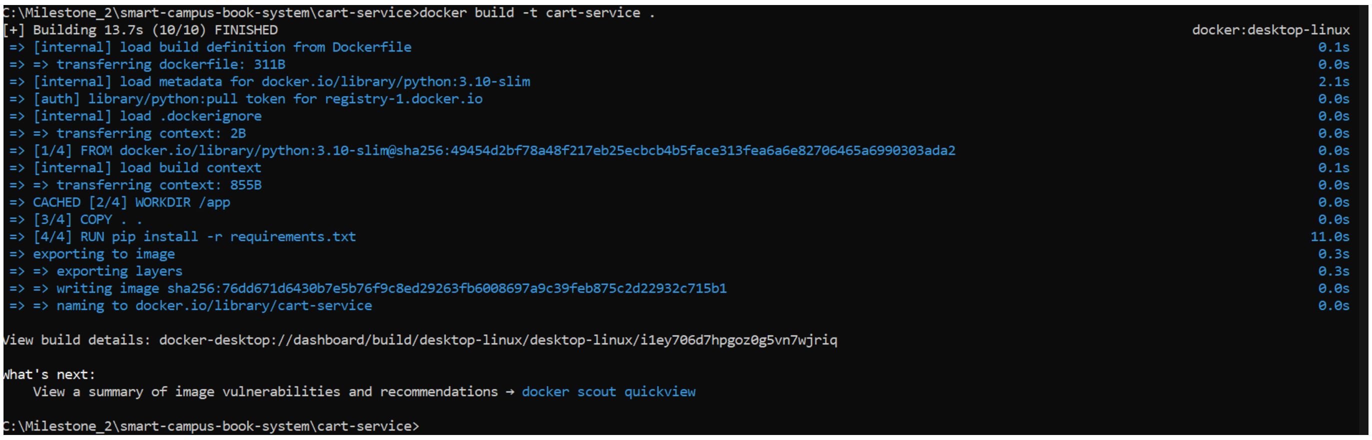


```
C:\Milestone_2\smart-campus-book-system\cart-service>dir
 Volume in drive C is OS
 Volume Serial Number is 169F-1F6E

 Directory of C:\Milestone_2\smart-campus-book-system\cart-service

2025/05/31  15:52    <DIR>          .
2025/05/31  14:20    <DIR>          ..
2025/05/31  15:48               455 app.py
2025/05/31  15:52               272 Dockerfile
2025/05/31  15:50                 7 requirements.txt
               3 File(s)            734 bytes
               2 Dir(s)  182 211 268 608 bytes free

C:\Milestone_2\smart-campus-book-system\cart-service>
```

Step 5: Build the Docker Image

Make sure you're inside the cart-service directory:

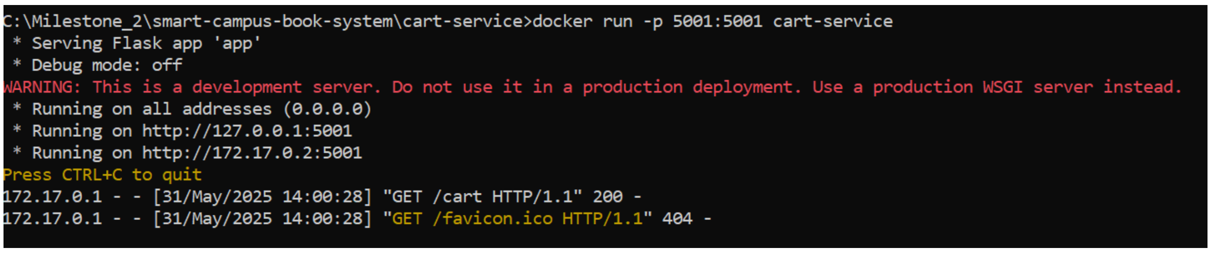
docker build -t cart-service .



```
C:\Milestone_2\smart-campus-book-system\cart-service>docker build -t cart-service .
[+] Building 13.7s (10/10) FINISHED                                                                           docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                          0.1s
 => => transferring dockerfile: 311B                                                                                          0.0s
 => [internal] load metadata for docker.io/library/python:3.10-slim                                                           2.1s
 => [auth] library/python:pull token for registry-1.docker.io                                                                 0.0s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [1/4] FROM docker.io/library/python:3.10-slim@sha256:49454d2bf78a48f217eb25ecbcb4b5face313fea6a6e82706465a6990303ada2     0.0s
 => [internal] load build context                                                                                             0.1s
 => => transferring context: 855B                                                                                             0.0s
 => CACHED [2/4] WORKDIR /app                                                                                                 0.0s
 => [3/4] COPY . .                                                                                                            0.0s
 => [4/4] RUN pip install -r requirements.txt                                                                                11.0s
 => exporting to image                                                                                                        0.3s
 => => exporting layers                                                                                                       0.3s
 => => writing image sha256:76dd671d6430b7e5b76f9c8ed29263fb6008697a9c39feb875c2d22932c715b1                                  0.0s
 => => naming to docker.io/library/cart-service                                                                               0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/i1ey706d7hpgoz0g5vn7wjriq

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\Milestone_2\smart-campus-book-system\cart-service>
```

Step 6: Run the Container

docker run -p 5001:5001 cart-service



```
C:\Milestone_2\smart-campus-book-system\cart-service>docker run -p 5001:5001 cart-service
 * 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://127.0.0.1:5001
 * Running on http://172.17.0.2:5001
Press CTRL+C to quit
172.17.0.1 - - [31/May/2025 14:00:28] "GET /cart HTTP/1.1" 200 -
172.17.0.1 - - [31/May/2025 14:00:28] "GET /favicon.ico HTTP/1.1" 404 -
```

- The browser or a client successfully hit /cart and got a 200 OK.

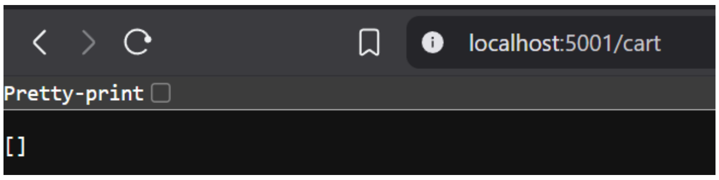
- The /favicon.ico 404 is normal — the browser checks for a website icon automatically.

Step 7: Test the Microservice

View cart (GET):

Visit http://localhost:5001/cart

You'll see an empty list [].

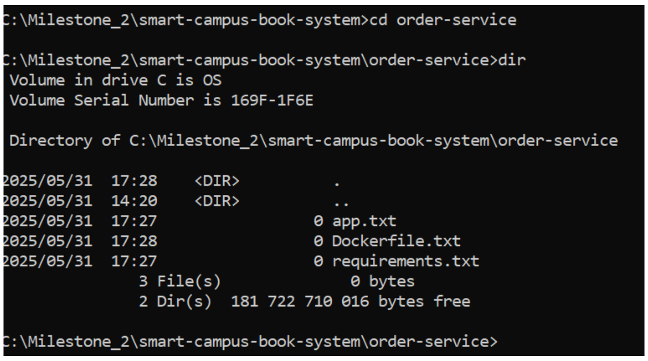


```
[]
```

order-service Microservice (Flask + Docker)

go on order-service

cd order-service



```
:\Milestone_2\smart-campus-book-system>cd order-service

:\Milestone_2\smart-campus-book-system\order-service>dir
 Volume in drive C is OS
 Volume Serial Number is 169F-1F6E

 Directory of C:\Milestone_2\smart-campus-book-system\order-service

025/05/31  17:28    <DIR>          .
025/05/31  14:20    <DIR>          ..
025/05/31  17:27                 0 app.txt
025/05/31  17:28                 0 Dockerfile.txt
025/05/31  17:27                 0 requirements.txt
               3 File(s)              0 bytes
               2 Dir(s)  181 722 710 016 bytes free

:\Milestone_2\smart-campus-book-system\order-service>
```

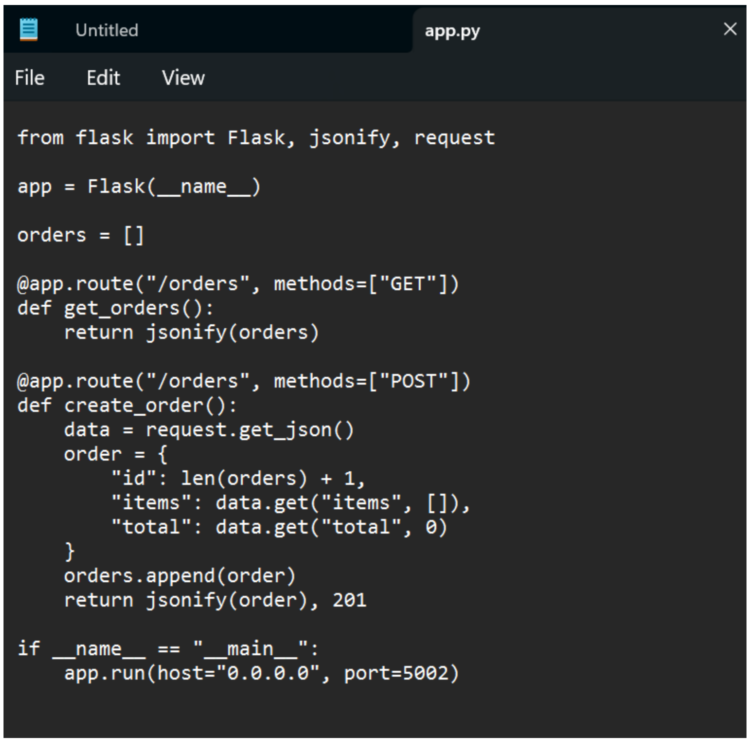
Create a new file called app.py and add this code in:

```
from flask import Flask, jsonify, request


app = Flask(__name__)


orders = []


@app.route("/orders", methods=["GET"])
def get_orders():
    return jsonify(orders)


@app.route("/orders", methods=["POST"])
def create_order():
    data = request.get_json()
    order = {
        "id": len(orders) + 1,
        "items": data.get("items", []),
        "total": data.get("total", 0)
    }
    orders.append(order)
```

return jsonify(order), 201

if __name__ == "__main__":

app.run(host="0.0.0.0", port=5002)

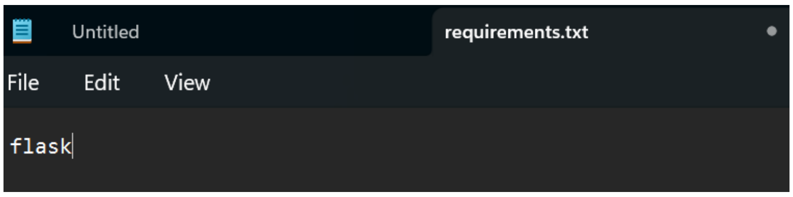


```
from flask import Flask, jsonify, request

app = Flask(__name__)

orders = []

@app.route("/orders", methods=["GET"])
def get_orders():
    return jsonify(orders)

@app.route("/orders", methods=["POST"])
def create_order():
    data = request.get_json()
    order = {
        "id": len(orders) + 1,
        "items": data.get("items", []),
        "total": data.get("total", 0)
    }
    orders.append(order)
    return jsonify(order), 201

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5002)
```

----

3. Create requirements.txt:

Flask



```
flask
```

----

4. Create Dockerfile:

# Use official Python base image

FROM python:3.9-slim

# Set working directory

WORKDIR /app

# Copy files

COPY . .

# Install dependencies

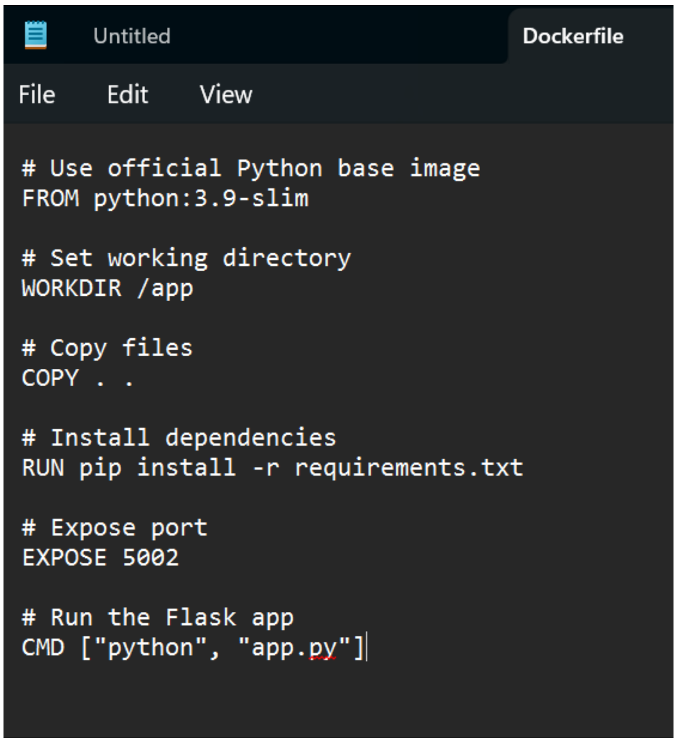
RUN pip install -r requirements.txt

# Expose port

EXPOSE 5002

# Run the Flask app

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



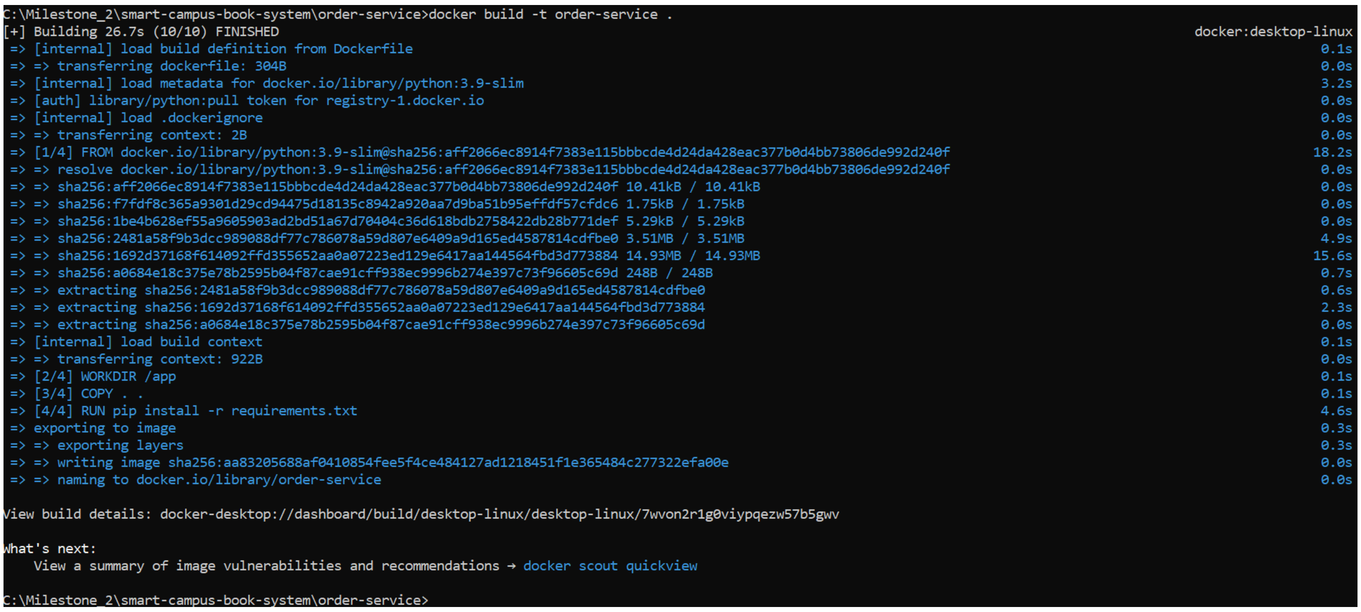
```
# Use official Python base image
FROM python:3.9-slim

# Set working directory
WORKDIR /app

# Copy files
COPY . .

# Install dependencies
RUN pip install -r requirements.txt

# Expose port
EXPOSE 5002

# Run the Flask app
CMD ["python", "app.py"]
```

----

5. Build the Docker image

Make sure you are inside order-service folder:

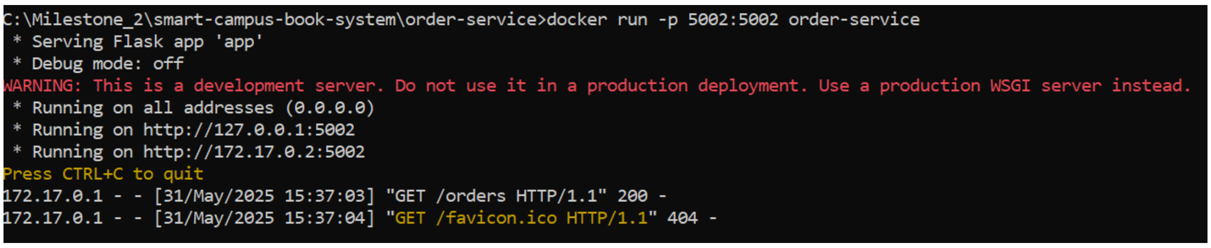
docker build -t order-service .



```
C:\Milestone_2\smart-campus-book-system\order-service>docker build -t order-service .
[+] Building 26.7s (10/10) FINISHED                                                                         docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                        0.1s
 => => transferring dockerfile: 304B                                                                                        0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                                          3.2s
 => [auth] library/python:pull token for registry-1.docker.io                                                               0.0s
 => [internal] load .dockerignore                                                                                           0.0s
 => => transferring context: 2B                                                                                             0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:aff2066ec8914f7383e115bbbcde4d24da428eac377b0d4bb73806de992d240f   18.2s
 => => resolve docker.io/library/python:3.9-slim@sha256:aff2066ec8914f7383e115bbbcde4d24da428eac377b0d4bb73806de992d240f    0.0s
 => => sha256:aff2066ec8914f7383e115bbbcde4d24da428eac377b0d4bb73806de992d240f 10.41kB / 10.41kB                            0.0s
 => => sha256:f7fdf8c365a9301d29cd94475d18135c8942a920aa7d9ba51b95effdf57cfdc6 1.75kB / 1.75kB                              0.0s
 => => sha256:1be4b628ef55a9605903ad2bd51a67d70404c36d618bdb2758422db28b771def 5.29kB / 5.29kB                              0.0s
 => => sha256:2481a58f9b3dcc989088df77c786078a59d807e6409a9d165ed4587814cdfbe0 3.51MB / 3.51MB                              4.9s
 => => sha256:1692d37168f614092ffd355652aa0a07223ed129e6417aa144564fbd3d773884 14.93MB / 14.93MB                           15.6s
 => => sha256:a0684e18c375e78b2595b04f87cae91cff938ec9996b274e397c73f96605c69d 248B / 248B                                  0.7s
 => => extracting sha256:2481a58f9b3dcc989088df77c786078a59d807e6409a9d165ed4587814cdfbe0                                   0.6s
 => => extracting sha256:1692d37168f614092ffd355652aa0a07223ed129e6417aa144564fbd3d773884                                   2.3s
 => => extracting sha256:a0684e18c375e78b2595b04f87cae91cff938ec9996b274e397c73f96605c69d                                   0.0s
 => [internal] load build context                                                                                           0.1s
 => => transferring context: 922B                                                                                           0.0s
 => [2/4] WORKDIR /app                                                                                                      0.1s
 => [3/4] COPY . .                                                                                                          0.1s
 => [4/4] RUN pip install -r requirements.txt                                                                               4.6s
 => exporting to image                                                                                                      0.3s
 => => exporting layers                                                                                                     0.3s
 => => writing image sha256:aa83205688af0410854fee5f4ce484127ad1218451f1e365484c277322efa00e                                0.0s
 => => naming to docker.io/library/order-service                                                                            0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/7wvon2r1g0viypqezw57b5gwv

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\Milestone_2\smart-campus-book-system\order-service>
```

6. Run the container:

docker run -p 5002:5002 order-service

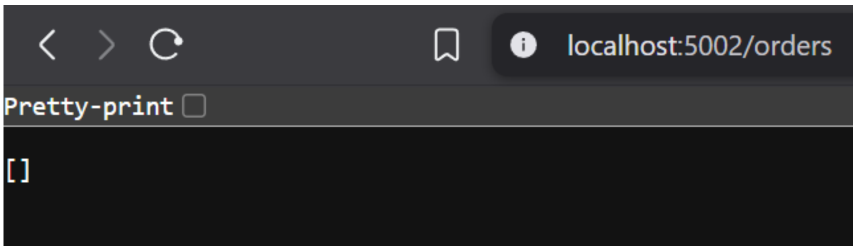


```
C:\Milestone_2\smart-campus-book-system\order-service>docker run -p 5002:5002 order-service
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5002
 * Running on http://172.17.0.2:5002
Press CTRL+C to quit
172.17.0.1 - - [31/May/2025 15:37:03] "GET /orders HTTP/1.1" 200 -
172.17.0.1 - - [31/May/2025 15:37:04] "GET /favicon.ico HTTP/1.1" 404 -
```

7. Test the service

In your browser, go to:

http://localhost:5002/orders



localhost:5002/orders

Pretty-print ☐

```
[]
```

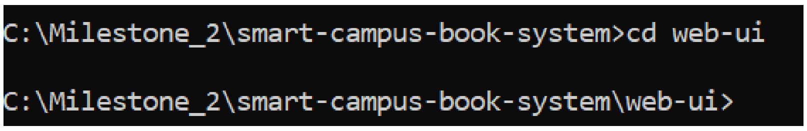
web-ui Microservice

let's now build the web-ui microservice that will act as the frontend for your Smart Campus Book Delivery System.

We'll create a simple HTML + Flask-based UI that connects to your existing microservices

Inter the "web-ui"

cd web-ui



```
C:\Milestone_2\smart-campus-book-system>cd web-ui

C:\Milestone_2\smart-campus-book-system\web-ui>
```

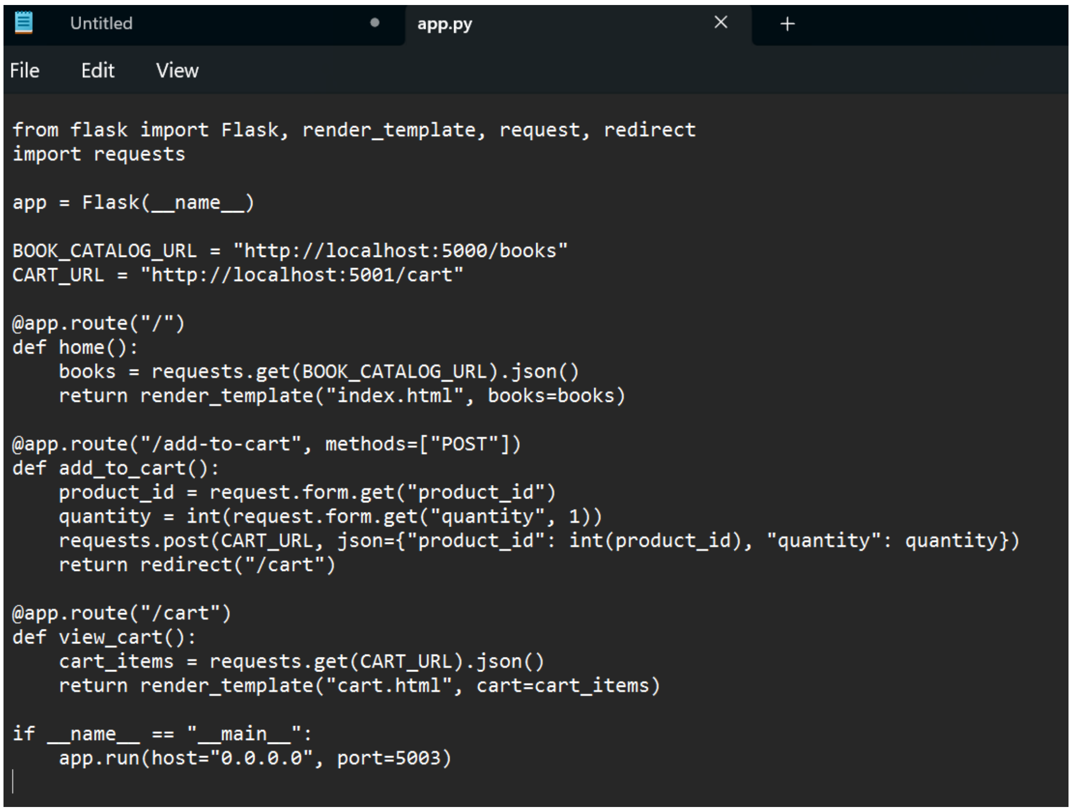
2. Create app.py

This will display book catalog, add to cart, and show cart content

The code:

```
from flask import Flask, render_template, request, redirect

import requests


app = Flask(__name__)


BOOK_CATALOG_URL = "http://book-catalog-service:5000/books"

CART_URL = "http://cart-service:5001/cart"


@app.route("/")

def home():

  books = requests.get(BOOK_CATALOG_URL).json()

  return render_template("index.html", books=books)


@app.route("/add-to-cart", methods=["POST"])

def add_to_cart():

  product_id = request.form.get("product_id")

  quantity = int(request.form.get("quantity", 1))
```

```
    requests.post(CART_URL, json={"product_id": int(product_id), "quantity": quantity})
    return redirect("/cart")


@app.route("/cart")
def view_cart():
    cart_items = requests.get(CART_URL).json()
    return render_template("cart.html", cart=cart_items)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5003)
```



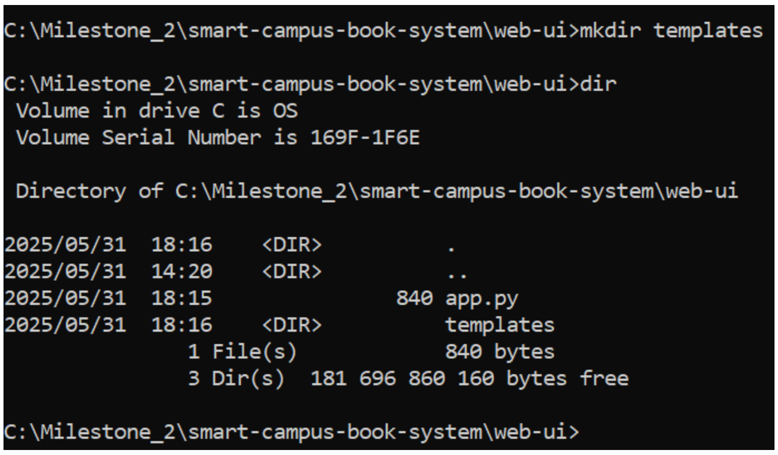
```
from flask import Flask, render_template, request, redirect
import requests

app = Flask(__name__)

BOOK_CATALOG_URL = "http://localhost:5000/books"
CART_URL = "http://localhost:5001/cart"

@app.route("/")
def home():
    books = requests.get(BOOK_CATALOG_URL).json()
    return render_template("index.html", books=books)

@app.route("/add-to-cart", methods=["POST"])
def add_to_cart():
    product_id = request.form.get("product_id")
    quantity = int(request.form.get("quantity", 1))
    requests.post(CART_URL, json={"product_id": int(product_id), "quantity": quantity})
    return redirect("/cart")

@app.route("/cart")
def view_cart():
    cart_items = requests.get(CART_URL).json()
    return render_template("cart.html", cart=cart_items)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5003)
```

----

3. Create HTML Templates

Create templates folder

mkdir templates

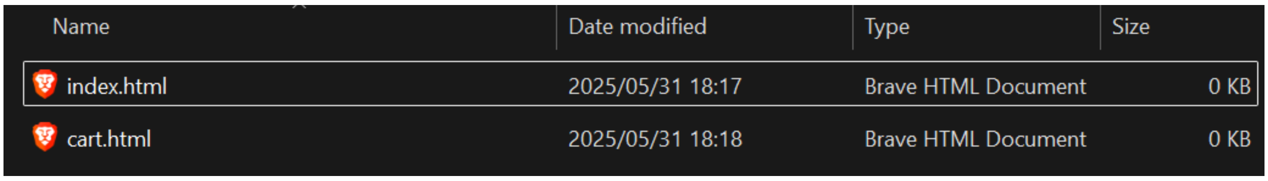


```
C:\Milestone_2\smart-campus-book-system\web-ui>mkdir templates

C:\Milestone_2\smart-campus-book-system\web-ui>dir
 Volume in drive C is OS
 Volume Serial Number is 169F-1F6E

 Directory of C:\Milestone_2\smart-campus-book-system\web-ui

2025/05/31  18:16    <DIR>          .
2025/05/31  14:20    <DIR>          ..
2025/05/31  18:15               840 app.py
2025/05/31  18:16    <DIR>          templates
               1 File(s)            840 bytes
               3 Dir(s)  181 696 860 160 bytes free

C:\Milestone_2\smart-campus-book-system\web-ui>
```

in the templates are 2 files:

- index.html

- cart.html

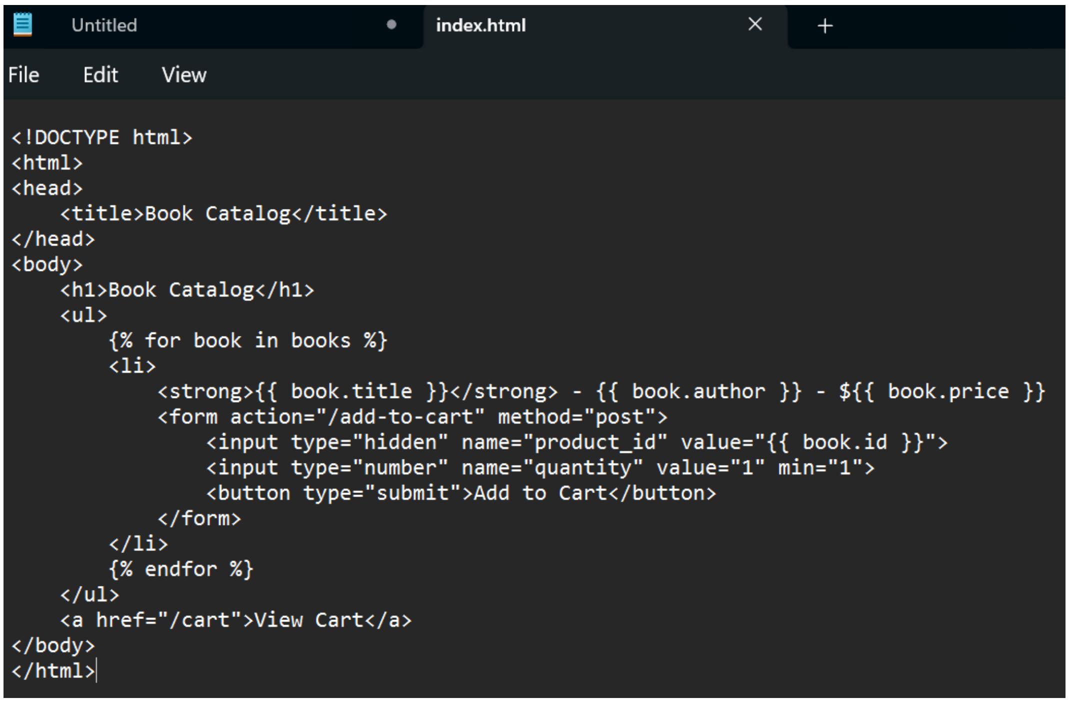


Name	Date modified	Type	Size
index.html	2025/05/31 18:17	Brave HTML Document	0 KB
cart.html	2025/05/31 18:18	Brave HTML Document	0 KB

in the index.html add this code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Book Catalog</title>
</head>
<body>
  <h1>Book Catalog</h1>
  <ul>
    {% for book in books %}
    <li>
      <strong>{{ book.title }}</strong> - {{ book.author }} - ${{ book.price }}
      <form action="/add-to-cart" method="post">
        <input type="hidden" name="product_id" value="{{ book.id }}">
        <input type="number" name="quantity" value="1" min="1">
```

```
            <button type="submit">Add to Cart</button>
          </form>
        </li>
        {% endfor %}
    </ul>
    <a href="/cart">View Cart</a>
</body>
</html>
```



```
<!DOCTYPE html>
<html>
<head>
    <title>Book Catalog</title>
</head>
<body>
    <h1>Book Catalog</h1>
    <ul>
        {% for book in books %}
        <li>
            <strong>{{ book.title }}</strong> - {{ book.author }} - ${{ book.price }}
            <form action="/add-to-cart" method="post">
                <input type="hidden" name="product_id" value="{{ book.id }}">
                <input type="number" name="quantity" value="1" min="1">
                <button type="submit">Add to Cart</button>
            </form>
        </li>
        {% endfor %}
    </ul>
    <a href="/cart">View Cart</a>
</body>
</html>
```

----

in the cart.html add this code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Your Cart</title>
</head>
<body>
    <h1>Cart Items</h1>
    <ul>
```

{% for item in cart %}

<li>Product ID: {{ item.product_id }} | Quantity: {{ item.quantity }}</li>

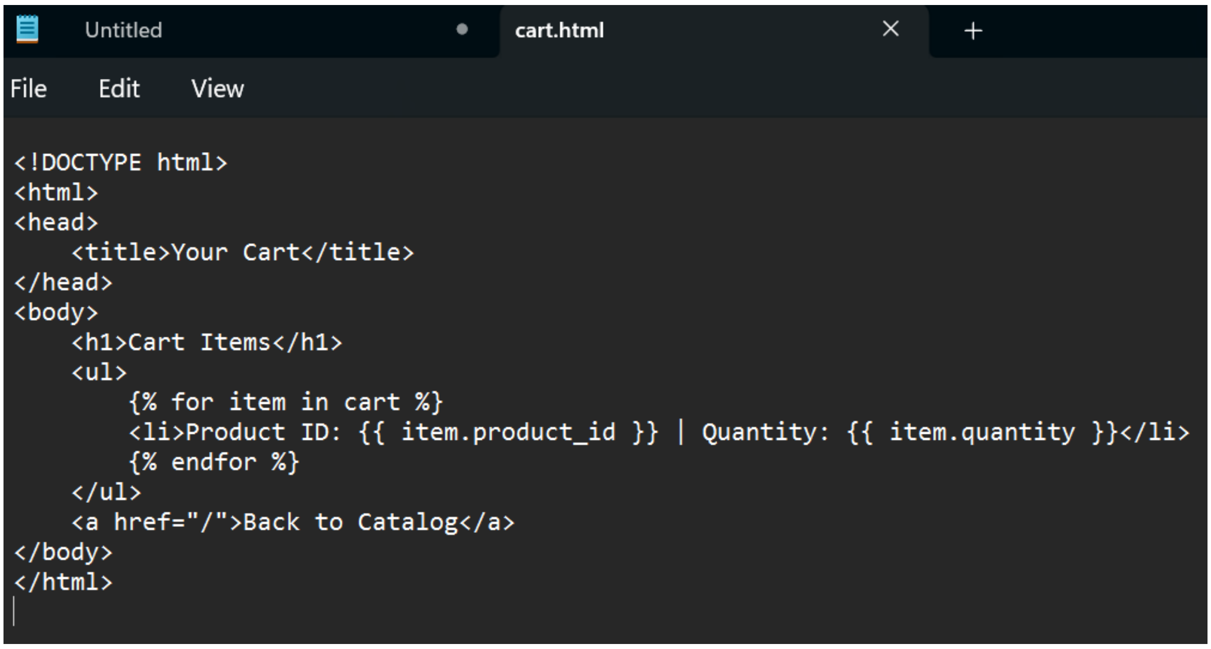
{% endfor %}

</ul>

<a href="/">Back to Catalog</a>

</body>

</html>



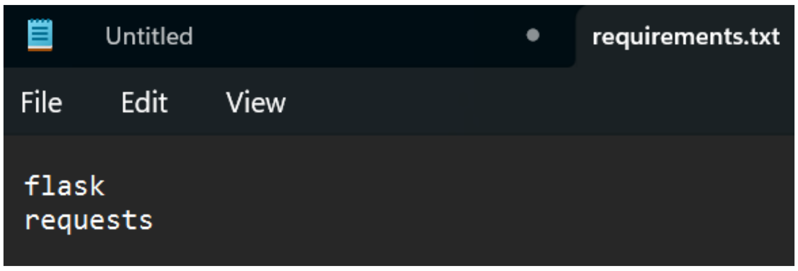
```
<!DOCTYPE html>
<html>
<head>
    <title>Your Cart</title>
</head>
<body>
    <h1>Cart Items</h1>
    <ul>
        {% for item in cart %}
        <li>Product ID: {{ item.product_id }} | Quantity: {{ item.quantity }}</li>
        {% endfor %}
    </ul>
    <a href="/">Back to Catalog</a>
</body>
</html>
```

----

back to the "" folder => in the requirements.txt add this code:

flask

requests



```
flask
requests
```

----

in the Dockerfile add this code:

FROM python:3.9-slim

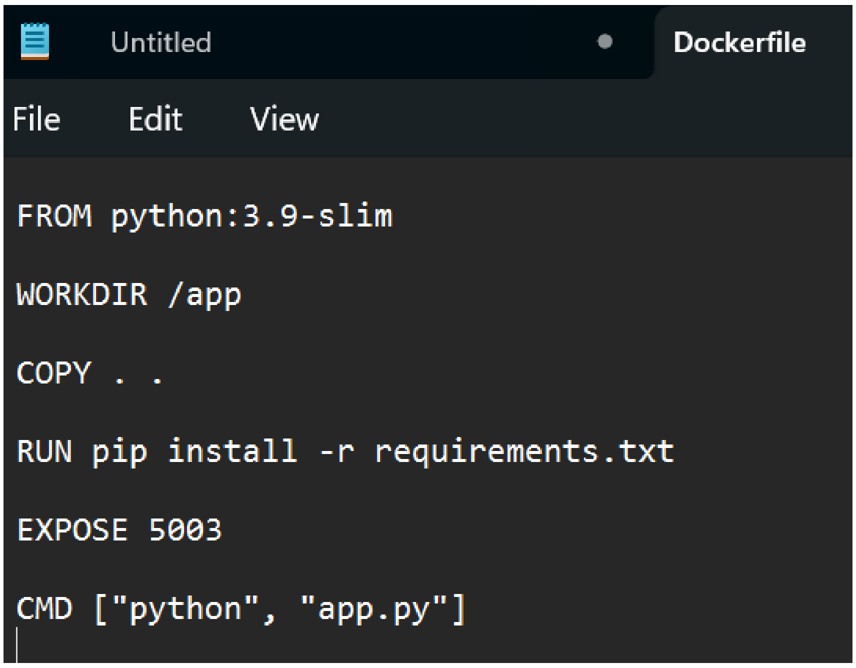
WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

EXPOSE 5003

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

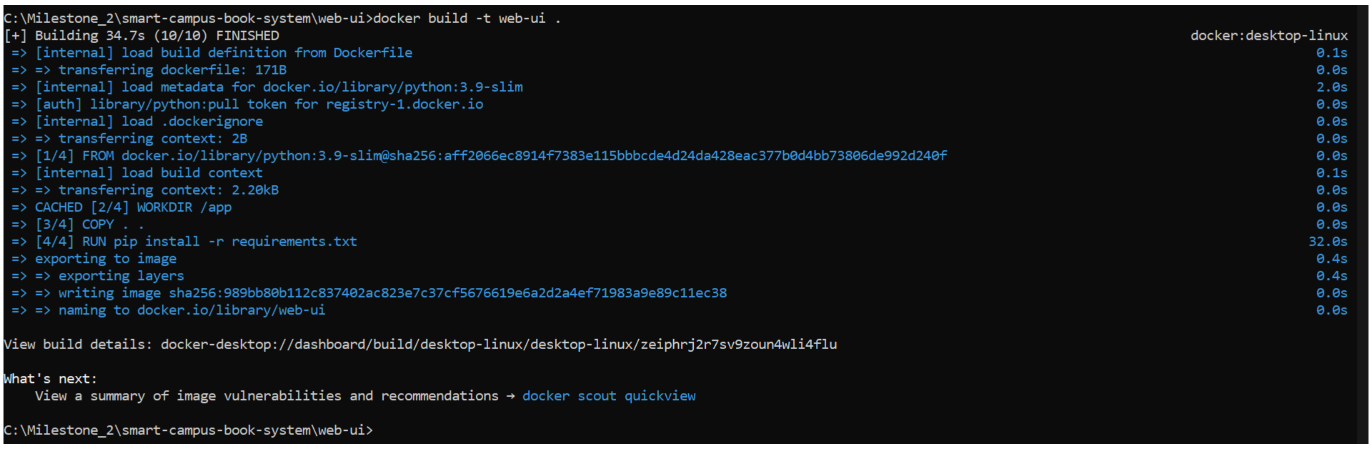


```
FROM python:3.9-slim

WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

EXPOSE 5003

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

----

6. Build the Docker image:

docker build -t web-ui .



```
C:\Milestone_2\smart-campus-book-system\web-ui>docker build -t web-ui .
[+] Building 34.7s (10/10) FINISHED                                                                                          docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                         0.1s
 => => transferring dockerfile: 171B                                                                                                         0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                                                           2.0s
 => [auth] library/python:pull token for registry-1.docker.io                                                                                0.0s
 => [internal] load .dockerignore                                                                                                            0.0s
 => => transferring context: 2B                                                                                                              0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:aff2066ec8914f7383e115bbbcde4d24da428eac377b0d4bb73806de992d240f                     0.0s
 => [internal] load build context                                                                                                            0.1s
 => => transferring context: 2.20kB                                                                                                          0.0s
 => CACHED [2/4] WORKDIR /app                                                                                                                0.0s
 => [3/4] COPY . .                                                                                                                           0.0s
 => [4/4] RUN pip install -r requirements.txt                                                                                               32.0s
 => exporting to image                                                                                                                       0.4s
 => => exporting layers                                                                                                                      0.4s
 => => writing image sha256:989bb80b112c837402ac823e7c37cf5676619e6a2d2a4ef71983a9e89c11ec38                                                 0.0s
 => => naming to docker.io/library/web-ui                                                                                                    0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/zeiphrj2r7sv9zoun4wli4flu

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\Milestone_2\smart-campus-book-system\web-ui>
```

7. Run the container:

docker run -p 5003:5003 web-ui

===

8. Test in Your Browser

http://localhost:5003/

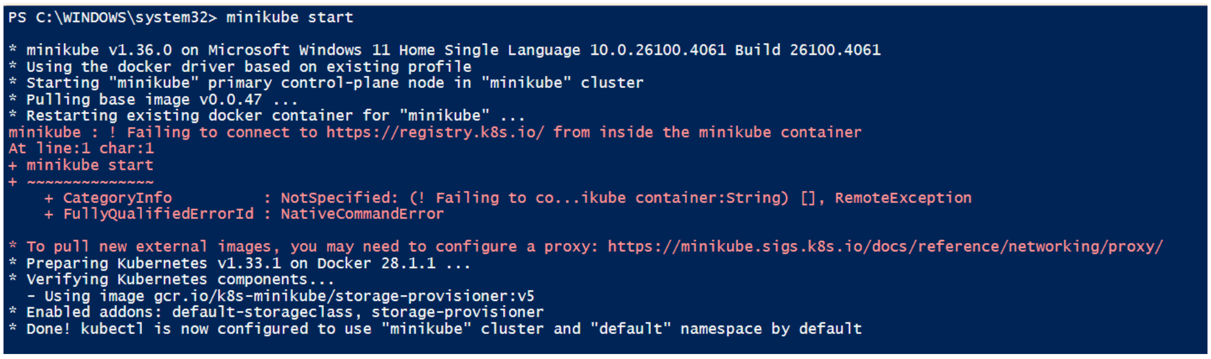
===

You should now see your book catalog and be able to add items to the cart and view them.

2. Kubernetes

Start Minikube on cmd

minikube start



```
PS C:\WINDOWS\system32> minikube start

* minikube v1.36.0 on Microsoft Windows 11 Home Single Language 10.0.26100.4061 Build 26100.4061
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.47 ...
* Restarting existing docker container for "minikube" ...
minikube : ! Failing to connect to https://registry.k8s.io/ from inside the minikube container
At line:1 char:1
+ minikube start
+ ~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (! Failing to co...ikube container:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError

* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
* Preparing Kubernetes v1.33.1 on Docker 28.1.1 ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

This launches a single-node Kubernetes cluster on your local machine.

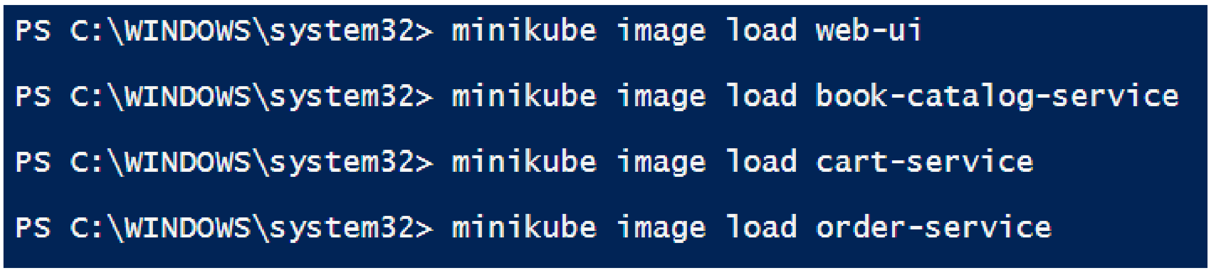
Load local Docker images into Minikube => For each microservice image run:

minikube image load web-ui

minikube image load book-catalog-service

minikube image load cart-service

minikube image load order-service



```
PS C:\WINDOWS\system32> minikube image load web-ui

PS C:\WINDOWS\system32> minikube image load book-catalog-service

PS C:\WINDOWS\system32> minikube image load cart-service

PS C:\WINDOWS\system32> minikube image load order-service
```

This copies your local image into Minikube's Docker environment so Kubernetes can use it in your deployments.

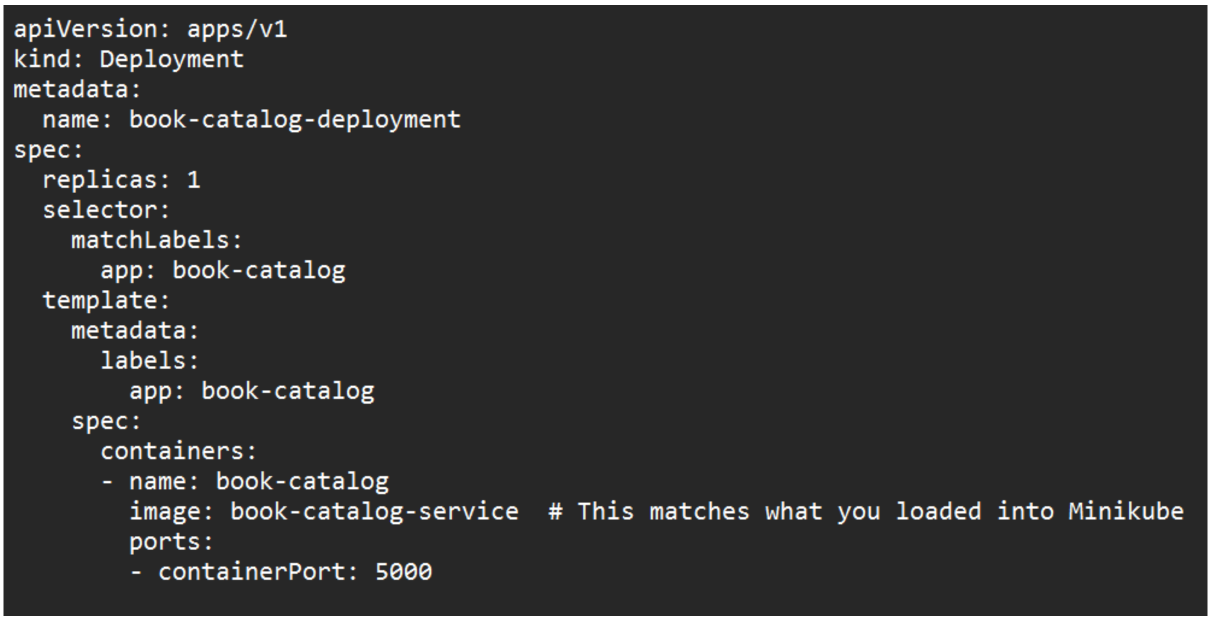
Create Kubernetes YAMLs for Each Microservice

Create a file called book-catalog-deployment.yaml with this content:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: book-catalog-deployment
spec:
 replicas: 1
 selector:
  matchLabels:
   app: book-catalog
 template:
  metadata:
   labels:
    app: book-catalog
  spec:
   containers:
   - name: book-catalog
    image: book-catalog-service  # This matches what you loaded into Minikube
```

ports:

- containerPort: 5000



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: book-catalog-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: book-catalog
  template:
    metadata:
      labels:
        app: book-catalog
    spec:
      containers:
      - name: book-catalog
        image: book-catalog-service  # This matches what you loaded into Minikube
        ports:
        - containerPort: 5000
```

Create a file called book-catalog-service.yaml:

apiVersion: v1

kind: Service

metadata:

 name: book-catalog-service

spec:

 selector:

  app: book-catalog

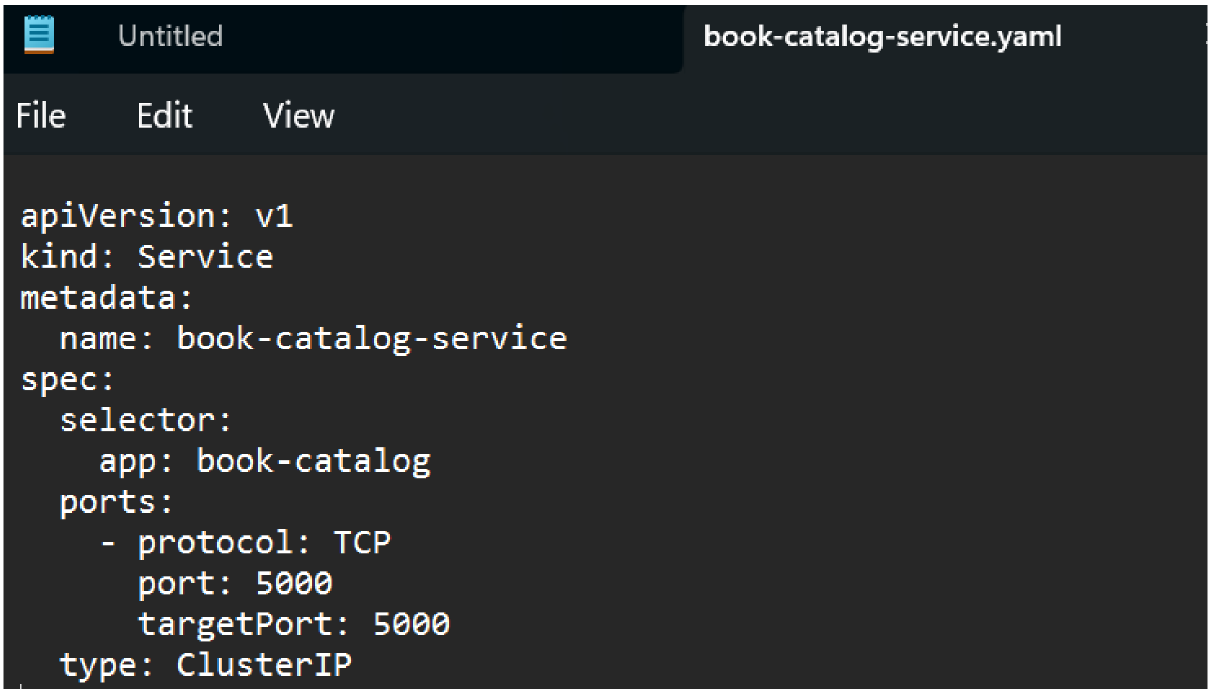
 ports:

  - protocol: TCP

   port: 5000

   targetPort: 5000

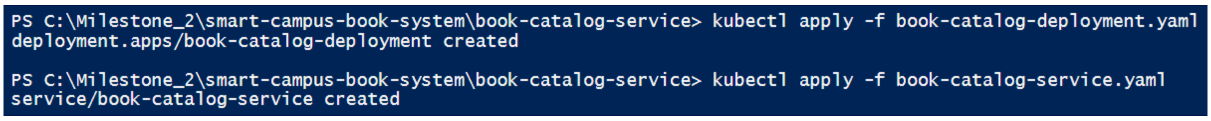
 type: ClusterIP



```yaml
apiVersion: v1
kind: Service
metadata:
  name: book-catalog-service
spec:
  selector:
    app: book-catalog
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
  type: ClusterIP
```

Once you've saved those files, apply them to your Kubernetes cluster using:

- **kubectl apply -f book-catalog-deployment.yaml**
- **kubectl apply -f book-catalog-service.yaml**



```
PS C:\Milestone_2\smart-campus-book-system\book-catalog-service> kubectl apply -f book-catalog-deployment.yaml
deployment.apps/book-catalog-deployment created

PS C:\Milestone_2\smart-campus-book-system\book-catalog-service> kubectl apply -f book-catalog-service.yaml
service/book-catalog-service created
```

Now do the same for:

- cart-service - Use cart-deployment.yaml and cart-service.yaml with port 5001
- order-service - Use order-deployment.yaml and order-service.yaml with port 5002
- web-ui - Use web-ui-deployment.yaml and web-ui-service.yaml with port 5003

like example:

Create a file called cart-deployment.yaml with this content:

apiVersion: apps/v1

kind: Deployment

metadata:

 name: cart-deployment

spec:

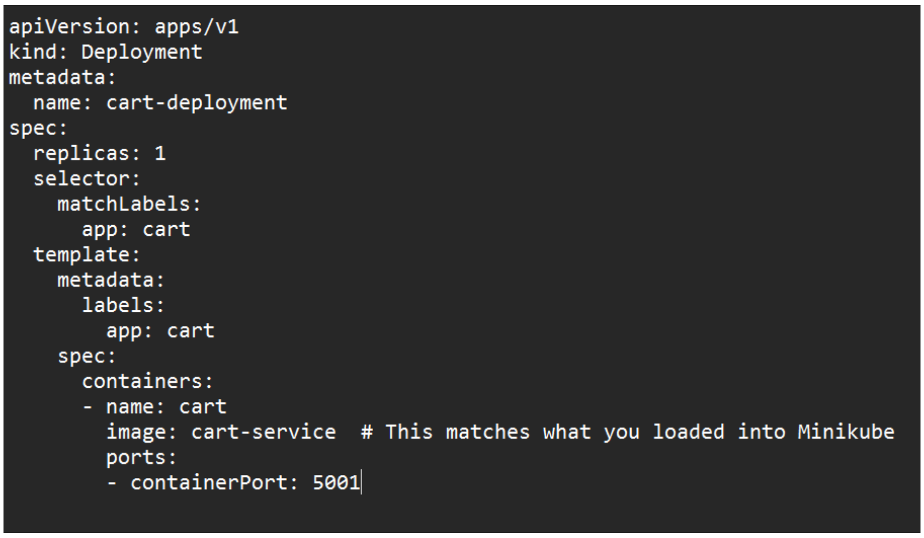
 replicas: 1

 selector:

  matchLabels:

   app: cart

```
 template:
  metadata:
   labels:
    app: cart
  spec:
   containers:
   - name: cart
    image: cart-service  # This matches what you loaded into Minikube
    ports:
    - containerPort: 5001
```

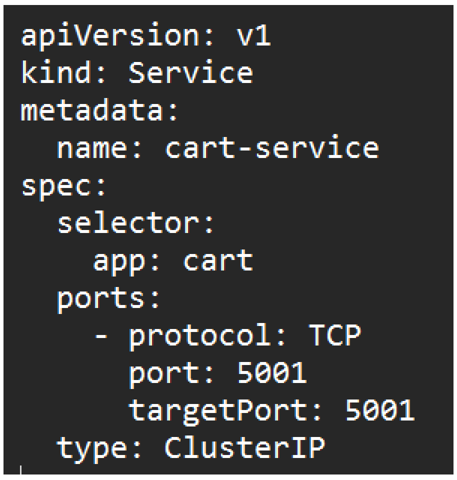


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cart-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: cart
  template:
    metadata:
      labels:
        app: cart
    spec:
      containers:
      - name: cart
        image: cart-service  # This matches what you loaded into Minikube
        ports:
        - containerPort: 5001
```

Create a file called cart-service.yaml with this content:

```
apiVersion: v1
kind: Service
metadata:
 name: cart-service
spec:
 selector:
  app: cart
 ports:
  - protocol: TCP
   port: 5001
   targetPort: 5001
```

type: ClusterIP

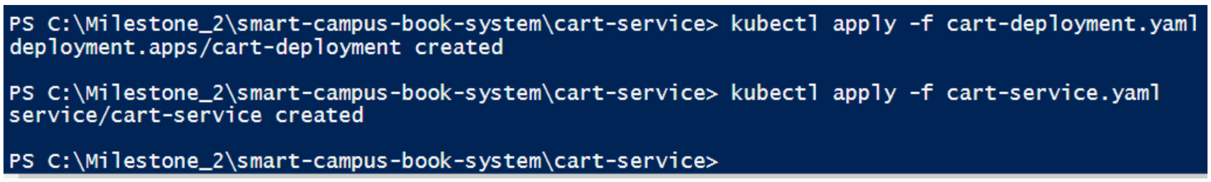


```
apiVersion: v1
kind: Service
metadata:
  name: cart-service
spec:
  selector:
    app: cart
  ports:
    - protocol: TCP
      port: 5001
      targetPort: 5001
  type: ClusterIP
```

Once you've saved those files, apply them to your Kubernetes cluster using:

kubectl apply -f cart-deployment.yaml

kubectl apply -f cart-service.yaml

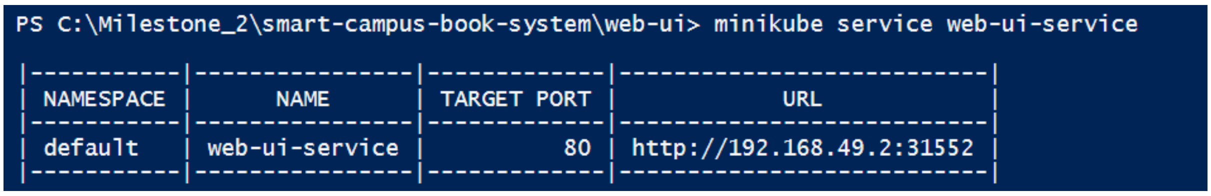


```
PS C:\Milestone_2\smart-campus-book-system\cart-service> kubectl apply -f cart-deployment.yaml
deployment.apps/cart-deployment created

PS C:\Milestone_2\smart-campus-book-system\cart-service> kubectl apply -f cart-service.yaml
service/cart-service created

PS C:\Milestone_2\smart-campus-book-system\cart-service>
```

After That: Access Web UI

Type this command:

minikube service web-ui-service



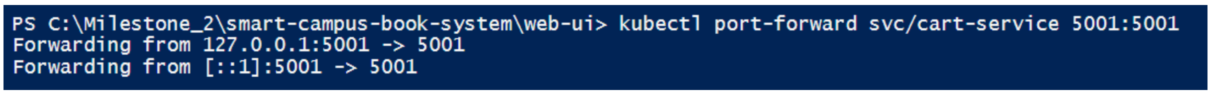
```
PS C:\Milestone_2\smart-campus-book-system\web-ui> minikube service web-ui-service

|-----------|----------------|-------------|----------------------------|
| NAMESPACE |      NAME      | TARGET PORT |            URL             |
|-----------|----------------|-------------|----------------------------|
| default   | web-ui-service |          80 | http://192.168.49.2:31552 |
|-----------|----------------|-------------|----------------------------|
```

This will open your browser with the correct Minikube IP and NodePort for the UI.

Test Inter-Service Communication in Kubernetes

Let's expose the cart-service on the local machine:

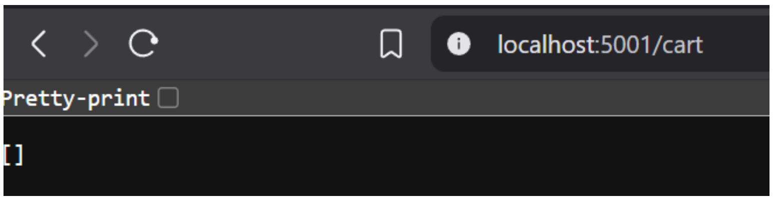
kubectl port-forward svc/cart-service 5001:5001



```
PS C:\Milestone_2\smart-campus-book-system\web-ui> kubectl port-forward svc/cart-service 5001:5001
Forwarding from 127.0.0.1:5001 -> 5001
Forwarding from [::1]:5001 -> 5001
```

Then open your browser and test:

http://localhost:5001/cart

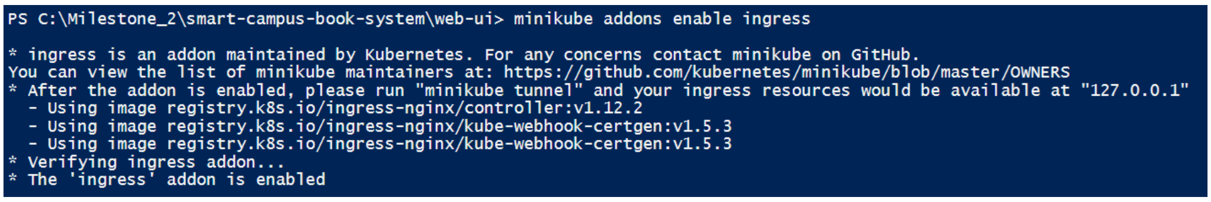


```
[]
```

If it returns a JSON (even an empty cart), it's working!

Enable Ingress Addon in Minikube

minikube addons enable ingress

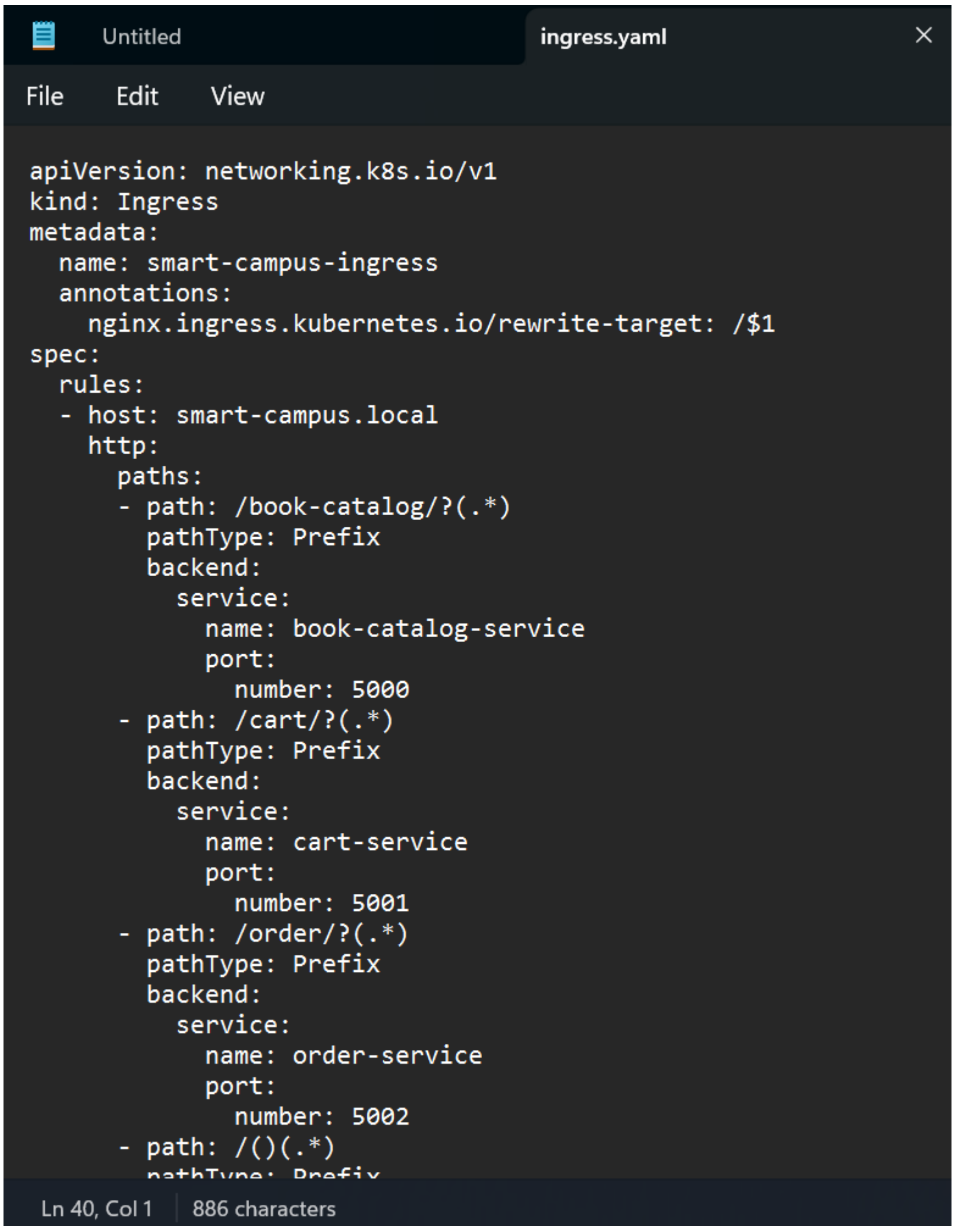


```
PS C:\Milestone_2\smart-campus-book-system\web-ui> minikube addons enable ingress

* ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
* After the addon is enabled, please run "minikube tunnel" and your ingress resources would be available at "127.0.0.1"
  - Using image registry.k8s.io/ingress-nginx/controller:v1.12.2
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.5.3
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.5.3
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

Create a file in your project root folder "smart-campus-book-system" called "ingress.yaml" with the following content:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: smart-campus-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
  - host: smart-campus.local
    http:
      paths:
```

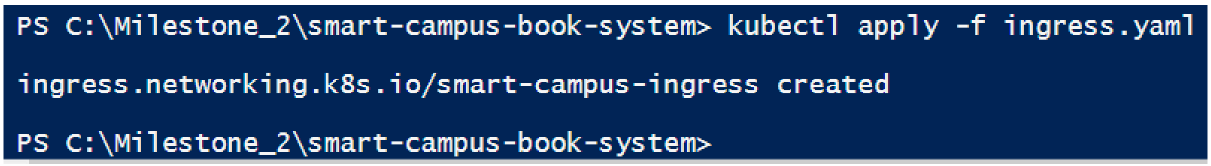
```
- path: /book-catalog/?(.*)
  pathType: Prefix
  backend:
    service:
      name: book-catalog-service
      port:
        number: 5000
- path: /cart/?(.*)
  pathType: Prefix
  backend:
    service:
      name: cart-service
      port:
        number: 5001
- path: /order/?(.*)
  pathType: Prefix
  backend:
    service:
      name: order-service
      port:
        number: 5002
- path: /()(.*)
  pathType: Prefix
  backend:
    service:
      name: web-ui-service
      port:
        number: 80
```



```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: smart-campus-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
  - host: smart-campus.local
    http:
      paths:
      - path: /book-catalog/?(.*)
        pathType: Prefix
        backend:
          service:
            name: book-catalog-service
            port:
              number: 5000
      - path: /cart/?(.*)
        pathType: Prefix
        backend:
          service:
            name: cart-service
            port:
              number: 5001
      - path: /order/?(.*)
        pathType: Prefix
        backend:
          service:
            name: order-service
            port:
              number: 5002
      - path: /()(.*)
```

Apply the Ingress by typing this code

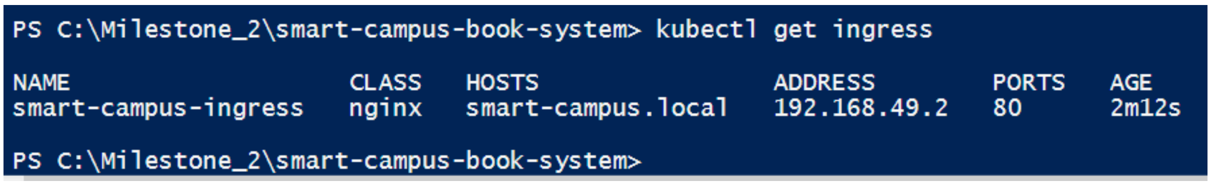
kubectl apply -f ingress.yaml



```
PS C:\Milestone_2\smart-campus-book-system> kubectl apply -f ingress.yaml
ingress.networking.k8s.io/smart-campus-ingress created
PS C:\Milestone_2\smart-campus-book-system>
```

To confirm type this

kubectl get ingress

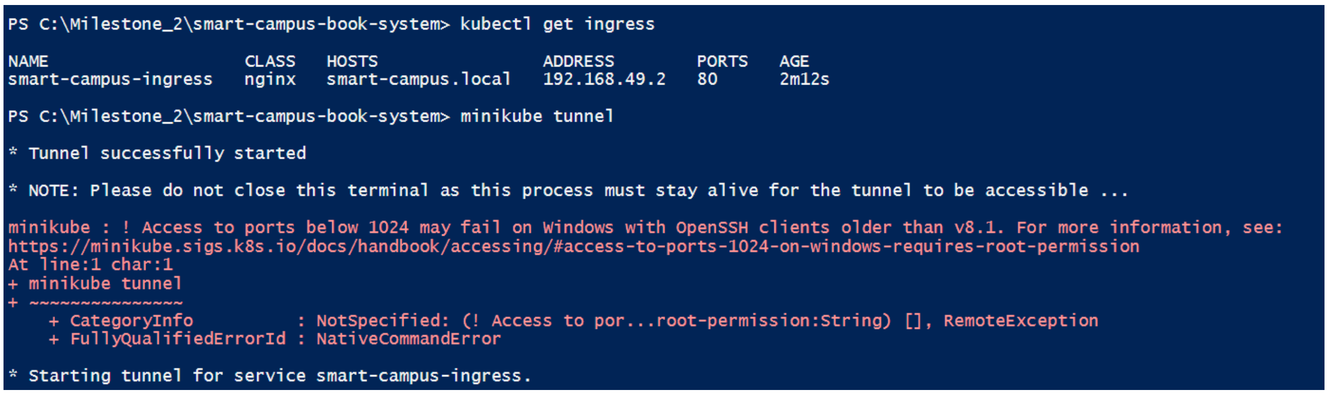


```
PS C:\Milestone_2\smart-campus-book-system> kubectl get ingress

NAME                   CLASS   HOSTS                ADDRESS        PORTS   AGE
smart-campus-ingress   nginx   smart-campus.local   192.168.49.2   80      2m12s

PS C:\Milestone_2\smart-campus-book-system>
```

Final Steps to Access It in Your Browser

Run the Ingress Tunnel

minikube tunnel



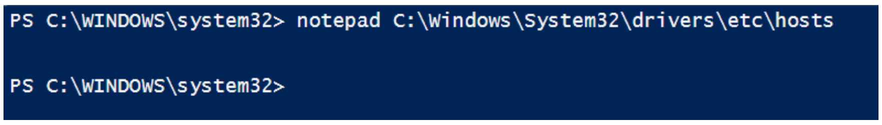
```
PS C:\Milestone_2\smart-campus-book-system> kubectl get ingress

NAME                   CLASS   HOSTS                ADDRESS        PORTS   AGE
smart-campus-ingress   nginx   smart-campus.local   192.168.49.2   80      2m12s

PS C:\Milestone_2\smart-campus-book-system> minikube tunnel

* Tunnel successfully started

* NOTE: Please do not close this terminal as this process must stay alive for the tunnel to be accessible ...

minikube : ! Access to ports below 1024 may fail on Windows with OpenSSH clients older than v8.1. For more information, see:
https://minikube.sigs.k8s.io/docs/handbook/accessing/#access-to-ports-1024-on-windows-requires-root-permission
At line:1 char:1
+ minikube tunnel
+ ~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (! Access to por...root-permission:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError

* Starting tunnel for service smart-campus-ingress.
```

This exposes Ingress on your host machine.

Make sure hosts file is updated

Open your hosts file in Notepad as Administrator:

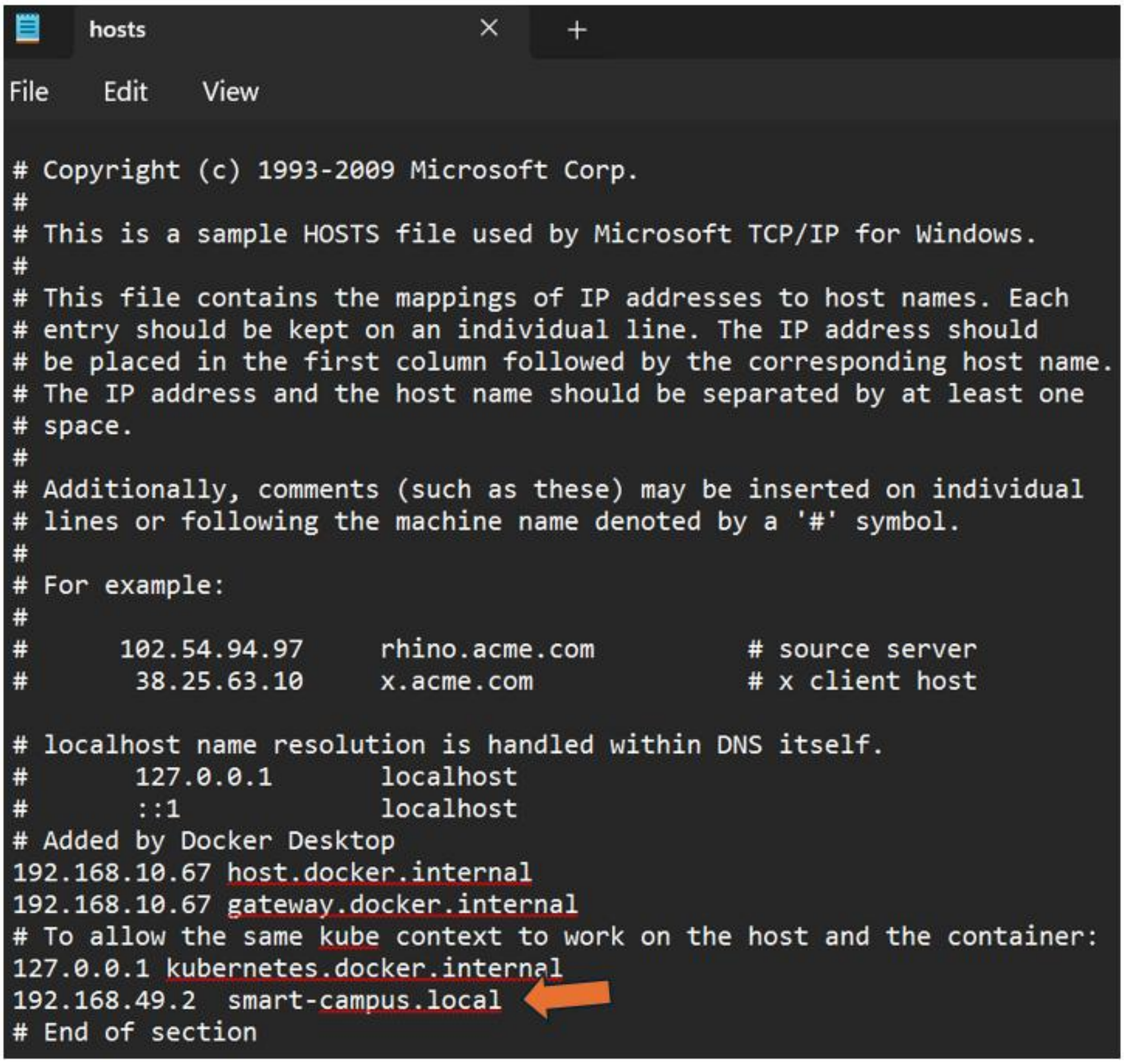
notepad C:\Windows\System32\drivers\etc\hosts



```
PS C:\WINDOWS\system32> notepad C:\Windows\System32\drivers\etc\hosts

PS C:\WINDOWS\system32>
```

Make sure you're still editing the hosts file as Administrator in Notepad. => Scroll to the bottom. => Add:

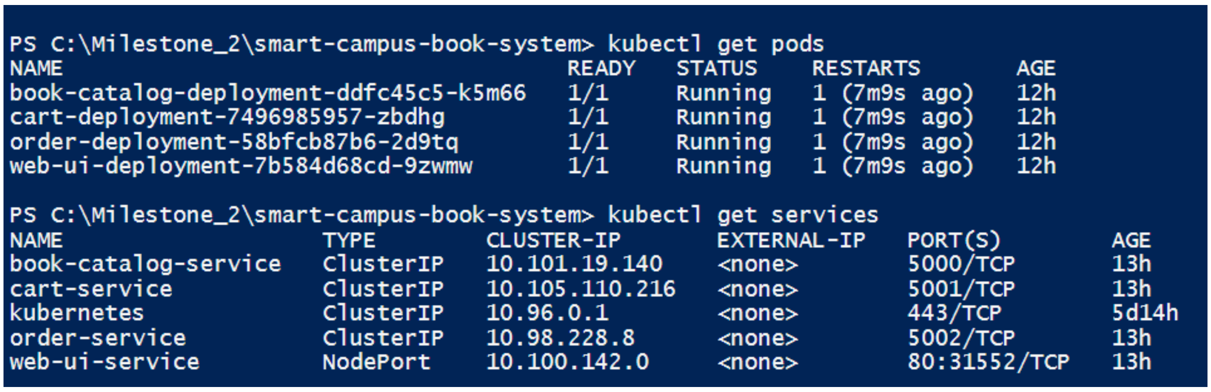
192.168.49.2 smart-campus.local



```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
# Added by Docker Desktop
192.168.10.67 host.docker.internal
192.168.10.67 gateway.docker.internal
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
192.168.49.2  smart-campus.local
# End of section
```

3. Scaling and Management

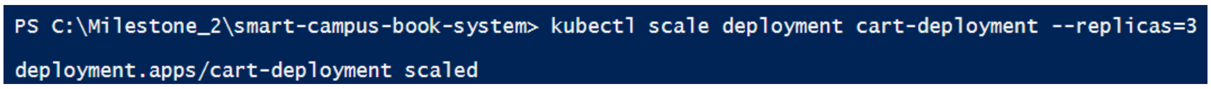
Let’s say you want to scale cart-deployment from 1 to 3 replicas



```
PS C:\Milestone_2\smart-campus-book-system> kubectl get pods
NAME                                  READY   STATUS    RESTARTS      AGE
book-catalog-deployment-ddfc45c5-k5m66   1/1     Running   1 (7m9s ago)   12h
cart-deployment-7496985957-zbdhg         1/1     Running   1 (7m9s ago)   12h
order-deployment-58bfcb87b6-2d9tq        1/1     Running   1 (7m9s ago)   12h
web-ui-deployment-7b584d68cd-9zwmw       1/1     Running   1 (7m9s ago)   12h

PS C:\Milestone_2\smart-campus-book-system> kubectl get services
NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
book-catalog-service   ClusterIP   10.101.19.140    <none>        5000/TCP       13h
cart-service           ClusterIP   10.105.110.216   <none>        5001/TCP       13h
kubernetes             ClusterIP   10.96.0.1        <none>        443/TCP        5d14h
order-service          ClusterIP   10.98.228.8      <none>        5002/TCP       13h
web-ui-service         NodePort    10.100.142.0     <none>        80:31552/TCP   13h
```

To scale any of your microservices you say 3 replica command and run:

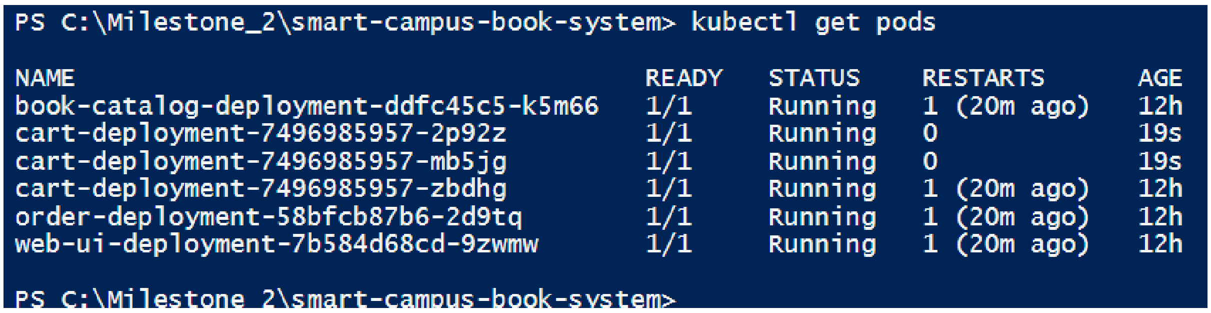
kubectl scale deployment cart-deployment --replicas=3



```
PS C:\Milestone_2\smart-campus-book-system> kubectl scale deployment cart-deployment --replicas=3
deployment.apps/cart-deployment scaled
```

Check That Pods Were Scaled by typing this command:

kubectl get pods



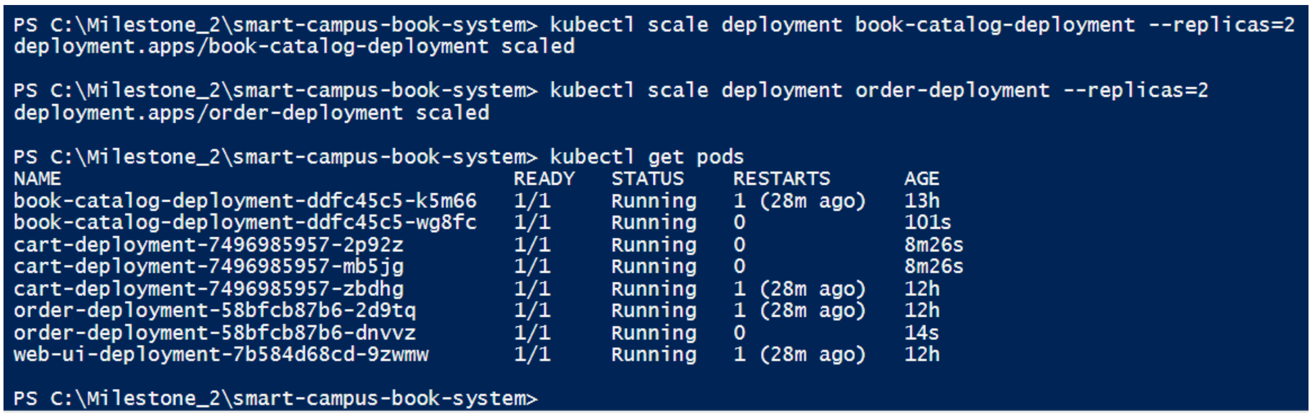
```
PS C:\Milestone_2\smart-campus-book-system> kubectl get pods

NAME                                    READY   STATUS    RESTARTS      AGE
book-catalog-deployment-ddfc45c5-k5m66   1/1     Running   1 (20m ago)   12h
cart-deployment-7496985957-2p92z         1/1     Running   0             19s
cart-deployment-7496985957-mb5jg         1/1     Running   0             19s
cart-deployment-7496985957-zbdhg         1/1     Running   1 (20m ago)   12h
order-deployment-58bfcb87b6-2d9tq        1/1     Running   1 (20m ago)   12h
web-ui-deployment-7b584d68cd-9zwmw       1/1     Running   1 (20m ago)   12h

PS C:\Milestone_2\smart-campus-book-system>
```

You should see extra pods with the same deployment name but different suffixes, showing the replicas have been created.

You can repeat the same command for other services, adjusting the replica count like you want:

kubectl scale deployment book-catalog-deployment --replicas=2

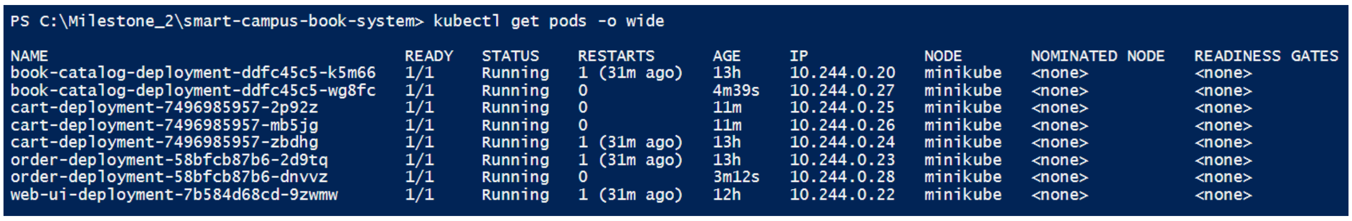
kubectl scale deployment order-deployment --replicas=2



```
PS C:\Milestone_2\smart-campus-book-system> kubectl scale deployment book-catalog-deployment --replicas=2
deployment.apps/book-catalog-deployment scaled

PS C:\Milestone_2\smart-campus-book-system> kubectl scale deployment order-deployment --replicas=2
deployment.apps/order-deployment scaled

PS C:\Milestone_2\smart-campus-book-system> kubectl get pods
NAME                                     READY   STATUS    RESTARTS      AGE
book-catalog-deployment-ddfc45c5-k5m66   1/1     Running   1 (28m ago)   13h
book-catalog-deployment-ddfc45c5-wg8fc   1/1     Running   0             101s
cart-deployment-7496985957-2p92z         1/1     Running   0             8m26s
cart-deployment-7496985957-mb5jg         1/1     Running   0             8m26s
cart-deployment-7496985957-zbdhg         1/1     Running   1 (28m ago)   12h
order-deployment-58bfcb87b6-2d9tq        1/1     Running   1 (28m ago)   12h
order-deployment-58bfcb87b6-dnvvz        1/1     Running   0             14s
web-ui-deployment-7b584d68cd-9zwmw       1/1     Running   1 (28m ago)   12h

PS C:\Milestone_2\smart-campus-book-system>
```

You can also use this command to monitor pod health and restart:

kubectl get pods -o wide



```
PS C:\Milestone_2\smart-campus-book-system> kubectl get pods -o wide

NAME                                     READY   STATUS    RESTARTS      AGE     IP            NODE       NOMINATED NODE   READINESS GATES
book-catalog-deployment-ddfc45c5-k5m66   1/1     Running   1 (31m ago)   13h     10.244.0.20   minikube   <none>           <none>
book-catalog-deployment-ddfc45c5-wg8fc   1/1     Running   0             4m39s   10.244.0.27   minikube   <none>           <none>
cart-deployment-7496985957-2p92z         1/1     Running   0             11m     10.244.0.25   minikube   <none>           <none>
cart-deployment-7496985957-mb5jg         1/1     Running   0             11m     10.244.0.26   minikube   <none>           <none>
cart-deployment-7496985957-zbdhg         1/1     Running   1 (31m ago)   13h     10.244.0.24   minikube   <none>           <none>
order-deployment-58bfcb87b6-2d9tq        1/1     Running   1 (31m ago)   13h     10.244.0.23   minikube   <none>           <none>
order-deployment-58bfcb87b6-dnvvz        1/1     Running   0             3m12s   10.244.0.28   minikube   <none>           <none>
web-ui-deployment-7b584d68cd-9zwmw       1/1     Running   1 (31m ago)   12h     10.244.0.22   minikube   <none>           <none>
```

- The scaled cart-deployment to 3 replicas and they're all Running.

- The scaled book-catalog-deployment and order-deployment 2 pods each now.

- Most pods have IP addresses in the range 10.244.x.x - assigned by Kubernetes networking, like Flannel or Calico in Minikube.

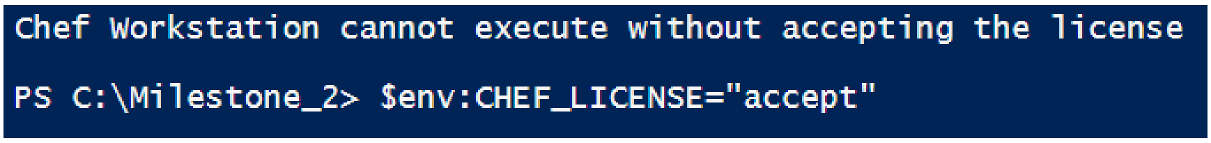
- All pods are scheduled on the same node "minikube", which is expected since I only have one VM running.

4. Basic Chef configurations

Accept the Chef License

permanently accept it for all commands:

$env:CHEF_LICENSE="accept"

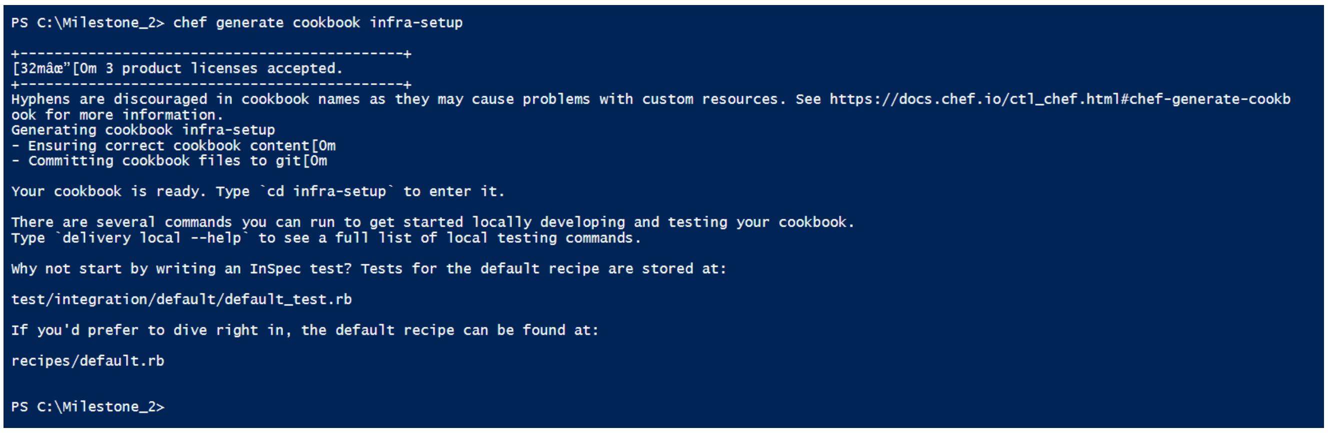


```
Chef Workstation cannot execute without accepting the license

PS C:\Milestone_2> $env:CHEF_LICENSE="accept"
```

Create the Cookbook

Open **PowerShell** and run:

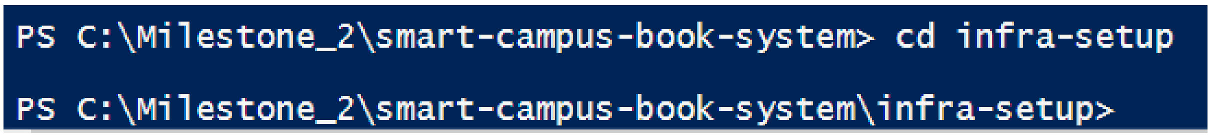
chef generate cookbook infra-setup



```
PS C:\Milestone_2> chef generate cookbook infra-setup

+---------------------------------------------+
[32mâœ"[0m 3 product licenses accepted.
+---------------------------------------------+
Hyphens are discouraged in cookbook names as they may cause problems with custom resources. See https://docs.chef.io/ctl_chef.html#chef-generate-cookb
ook for more information.
Generating cookbook infra-setup
- Ensuring correct cookbook content[0m
- Committing cookbook files to git[0m

Your cookbook is ready. Type `cd infra-setup` to enter it.

There are several commands you can run to get started locally developing and testing your cookbook.
Type `delivery local --help` to see a full list of local testing commands.

Why not start by writing an InSpec test? Tests for the default recipe are stored at:

test/integration/default/default_test.rb

If you'd prefer to dive right in, the default recipe can be found at:

recipes/default.rb

PS C:\Milestone_2>
```

Go to the directory that is created:

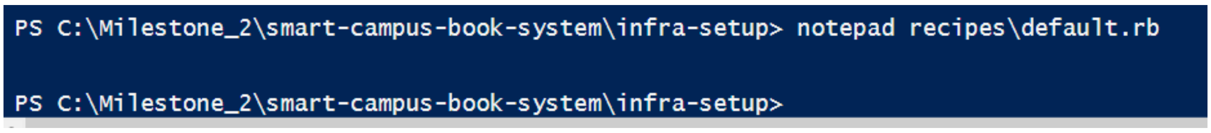
cd infra-setup



```
PS C:\Milestone_2\smart-campus-book-system> cd infra-setup

PS C:\Milestone_2\smart-campus-book-system\infra-setup>
```

Run this in PowerShell to open it with Notepad:

notepad recipes\default.rb



```
PS C:\Milestone_2\smart-campus-book-system\infra-setup> notepad recipes\default.rb

PS C:\Milestone_2\smart-campus-book-system\infra-setup>
```

Add the following code into default.rb:

```
# Cookbook:: infra-setup

# Recipe:: default


# Update the package list

execute 'apt_update' do

  command 'apt-get update'

end


# Install Docker

package 'docker.io' do

  action :install

end


# Install Git

package 'git' do

  action :install

end


# Install Python3 and pip

package 'python3' do

  action :install

end


package 'python3-pip' do

  action :install
```

```
end


# Disable UFW firewall
execute 'disable_ufw' do
  command 'ufw disable'
  only_if 'which ufw'
end
```

Then save the work

Upload the Cookbook to Chef Server

knife cookbook upload infra-setup

===

This command uploads the infra-setup cookbook to the Chef server.

Add the Recipe to the Node's Run List

knife node run-list add smartcampus-node 'recipe[infra-setup]'

===

This tells the node to run your recipe when chef-client runs.

References

Casperson, M., 2022. The difference between ClusterIP, NodePort, and LoadBalancer Kubernetes services. 14 November, pp. https://octopus.com/blog/difference-clusterip-nodeport-loadbalancer-kubernetes.

ElMalatawey, S., 2016. Bootstrap a Single-node VSAN Cluster under Running vCenter Server. 23 July, pp. https://vmusketeers.com/2016/07/23/bootstrap-a-single-node-vsan-cluster-under-running-vcenter-server/.

Heidi, E., 2020. configuration-management-101-writing-chef-recipes. 12 March, pp. https://www.digitalocean.com/community/tutorials/configuration-management-101-writing-chef-recipes.

Kardgar, B., 2023. What Is Kubernetes Deployment And How To Use It?. 08 Apr, pp. https://behdadk.medium.com/what-is-kubernetes-deployment-and-how-to-use-it-212210e5ad94.

Menachem, 2024. What Are Kubernetes Pod Statuses and 4 Ways to Monitor Them. 14 Feb, pp. https://komodor.com/learn/what-are-kubernetes-pod-statuses-and-4-ways-to-monitor-them/.

Menachem, G., 2023. How to Scale Kubernetes Pods with Kubectl Scale Deployment. 08 Aug, pp. https://komodor.com/learn/kubectl-scale-deployment-the-basics-and-a-quick-tutorial/.

Unk, 2023. Mastering Ingress in the UI: Elevating your app visibility. 3 Nov, pp. https://www.ibm.com/think/insights/mastering-ingress-in-the-ui-elevating-your-app-visibility.