AI 510 Artificial Intelligence of Cloud Computing
HOS02A Introduction to Docker
12/23/2022 Reviewed by Alekhya Malla
06/10/2024 Reviewed by Naveena Moddu
06/12/2024 Reviewed by Anh Nguyen
School of Technology and Computing (STC) @City University of Seattle (CityU)

Before You Start

- The directory path shown in the screenshots may be different from yours.
- Some steps are not explained in the tutorial**.** If you are not sure what to do:
  - Consult the resources listed below.
  - If you cannot solve the problem after a few tries, please contact the student worker through the MS Teams course channel.

Learning Outcomes

- Students will be able to learn:
  - Introduction to Docker
  - Docker Installation
  - Running Flask backend App as a Docker container

Resource

- Docker, (n.d), Docker documentation, https://docs.docker.com/get-started/
- Docker, (2020), How to Get Started with Docker, https://www.youtube.com/watch?v=iqqDU2crIEQ&t=30s
- Docker, (n.d), Docker 101 Tutorial, https://www.docker.com/101-tutorial/
- Tutorialspoint(n.d), Docker Tutorial, https://www.tutorialspoint.com/docker/index.htm
- Flask, (n.d), Flask Documentation, https://flask.palletsprojects.com/en/3.0.x/quickstart/
- Tutorialspoint, (n.d), Flask Tutorial, https://www.tutorialspoint.com/flask/index.htm

Introduction to Docker

Docker

Docker is an open-source platform that uses OS-level virtualization that enables developers to package applications into containers. Applications can be built, run, managed, and distributed with ease using this software platform.

Containers

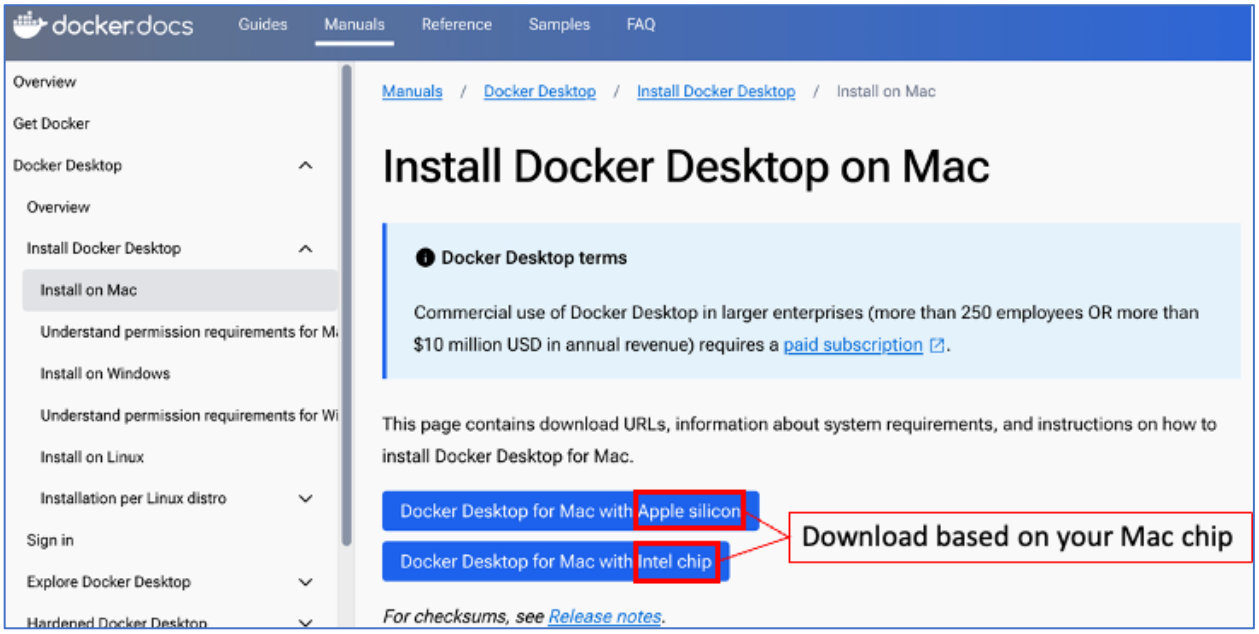
Containers are an isolated environment for running an application and standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run code in any environment. Docker Engine is software that hosts containers.

Docker Image

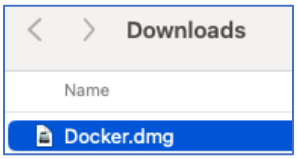
Docker image is a template containing the application and all dependencies required to run applications on Docker.

Installing Docker – Mac OSX

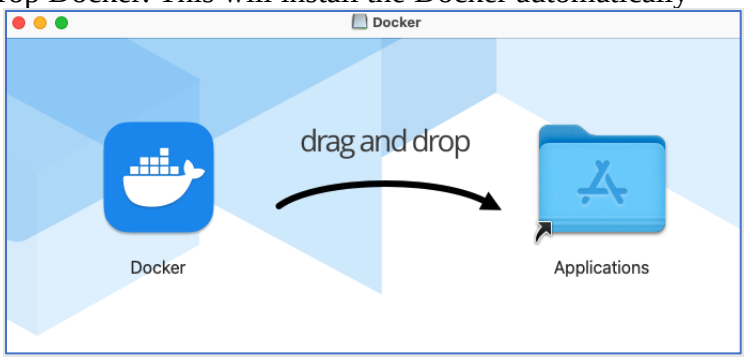
Step 1) Visit https://docs.docker.com/desktop/install/mac-install/ to download the appropriate installer. (ex. M series chips – Apple Silicon, Intel base – Intel chip)



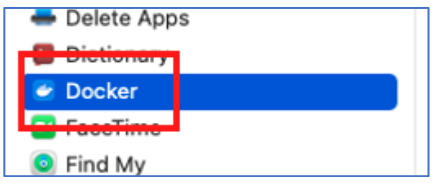
Step 2) Go to the location of download folder and double click Docker.dmg



Step 3) Drag and Drop Docker. This will install the Docker automatically



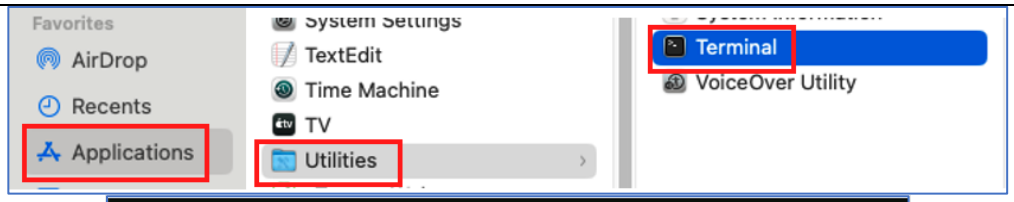
Step 4) If the Docker application doesn't start itself, open application folder and double click on the Docker icon



Step 5) Follow the screen to create account and sign in. You may need to click “re-authorize application” on the top of Docker desktop application or give “privileged access” on pop up with OSX account password.

Step 6) Open terminal application from application folder. Type the following command in Terminal to see if docker version prints.

```
docker version
```




```
(base) yaredshewarade@yared ~ % docker version
Client:
 Cloud integration: v1.0.28
 Version:           20.10.17
 API version:       1.41
 Go version:        go1.17.11
 Git commit:        100c701
 Built:             Mon Jun  6 23:04:45 2022
 OS/Arch:           darwin/arm64
 Context:           default
 Experimental:      true
```

Installing Docker – Windows

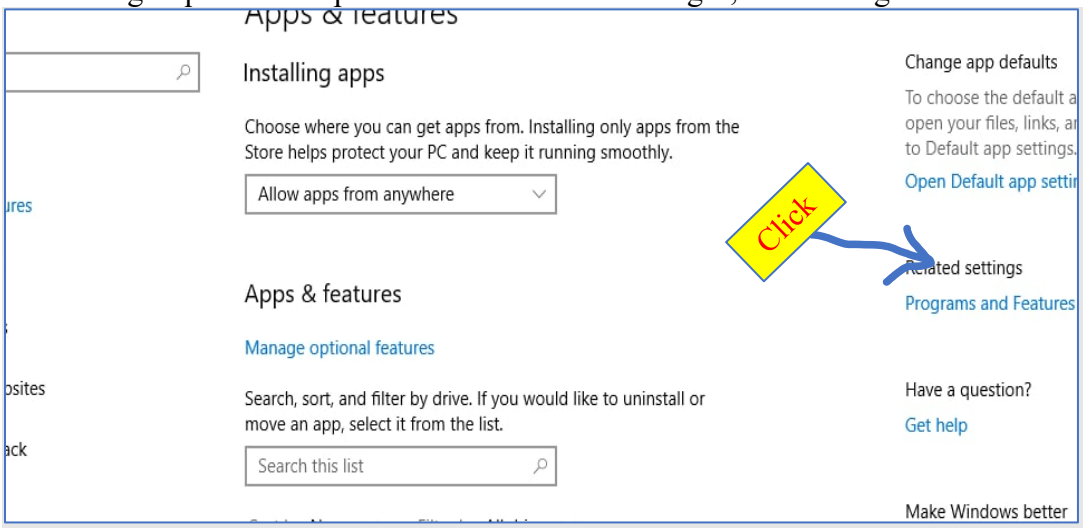
Step 1) Visit https://docs.docker.com/desktop/install/windows-install/ and click "Docker Desktop for Windows" to download installer.

Note: There is a list of system requirements. Be sure to check the list to see if the working system meets the requirements.

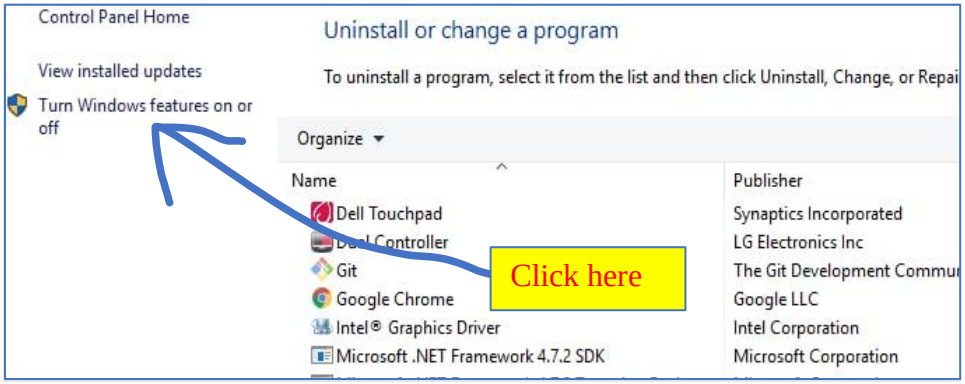
Step 2) Hyper-V should be enabled on Windows 10

- Right click on Windows button and select "Apps and Features"

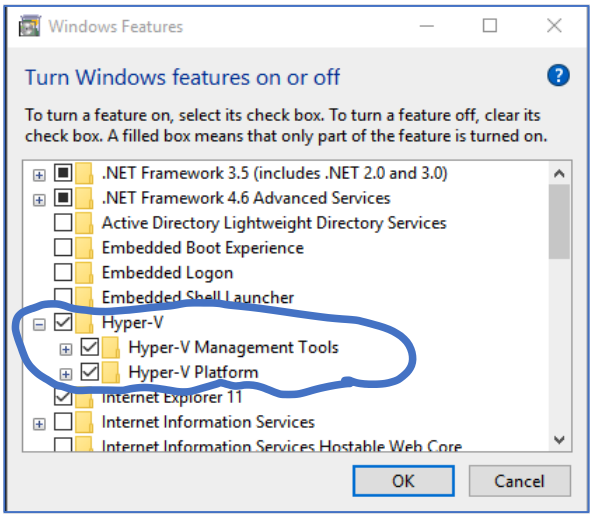
- On the right quick menu panel under "Related Settings", click "Programs and Features"



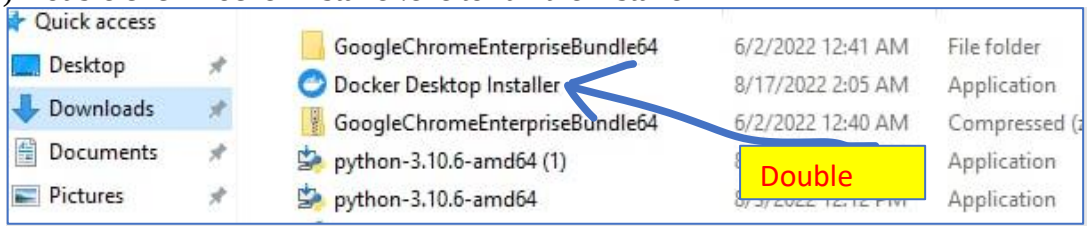
- Select “Turn Windows Feature on or off”



- Select Hyper-V and click OK

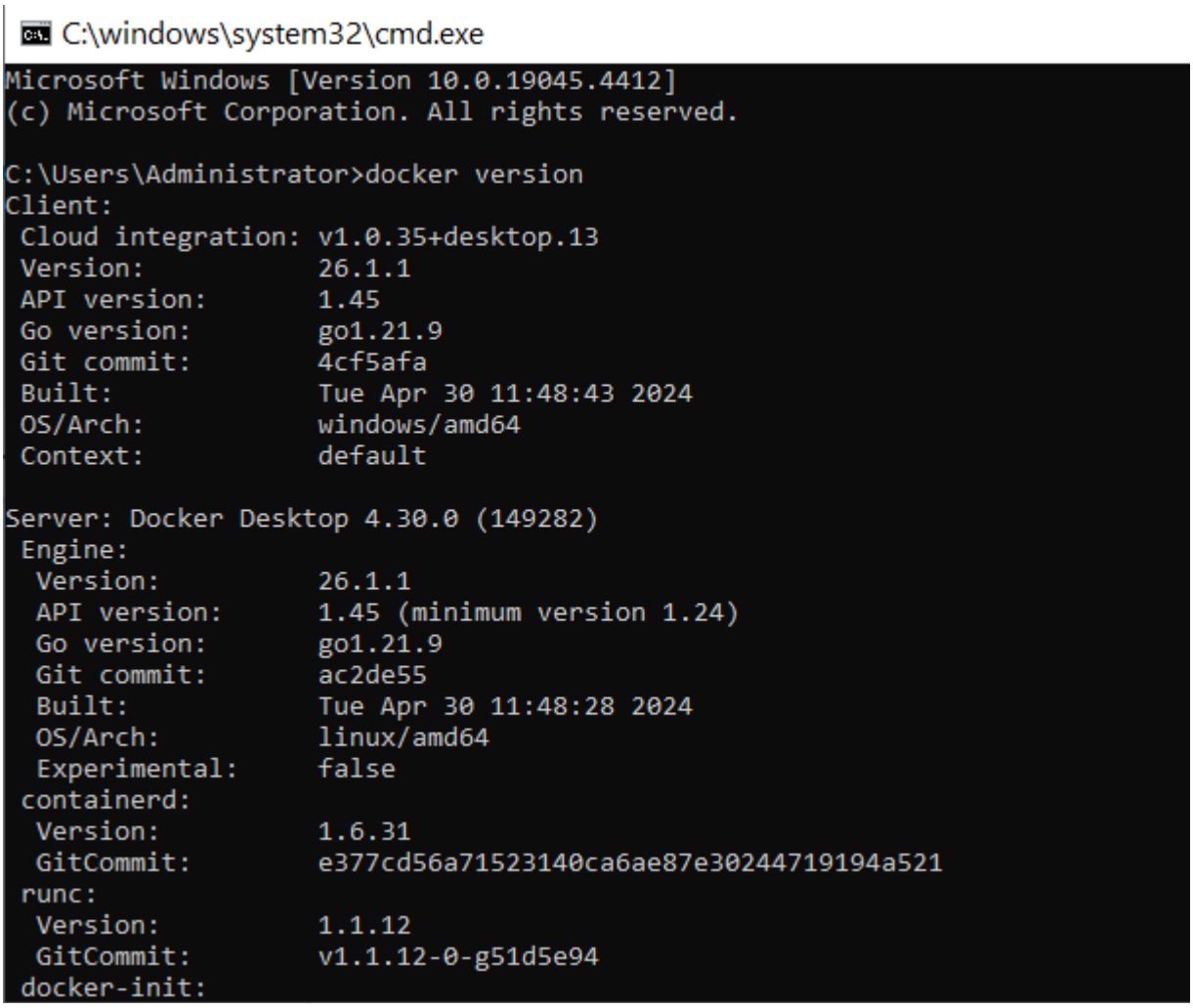


Step 3) Double click Docker Installer.exe to run the installer



Step 4) When prompted, ensure to use the “WSL 2” instead of Hyper-V option on the configuration. Regardless of your system, there will be a default selection already inplaced beside “WSL 2” and ensure to choose “WSL 2”. When the installation is successful, click “Close” to complete installation process.

Open command prompt and check the docker version.



Note: If Docker desktop takes long time to start after default installation try the following commands in the PowerShell application (ex. Start button > type "powershell" > click "Windows PowerShell")

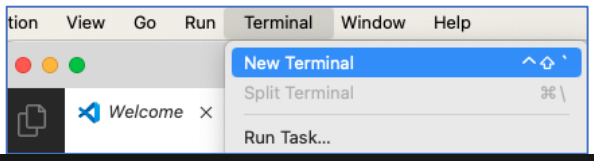
```
wsl --update
wsl --list --online
wsl --install -d Ubuntu-20.04
```

After wsl installation, close all Docker-related applications, and restart the Docker desktop application.

Docker practice

!!!During practice!!!, take **screenshots** for **Docker practice - steps 7, 8, 9, 14 and save them** into your local repository.

Step 1) Open VSCode and start a terminal panel. Navigate to your working directory (ex. Use cd command), and create a new folder called “flask_docker” in your Module02 folder.



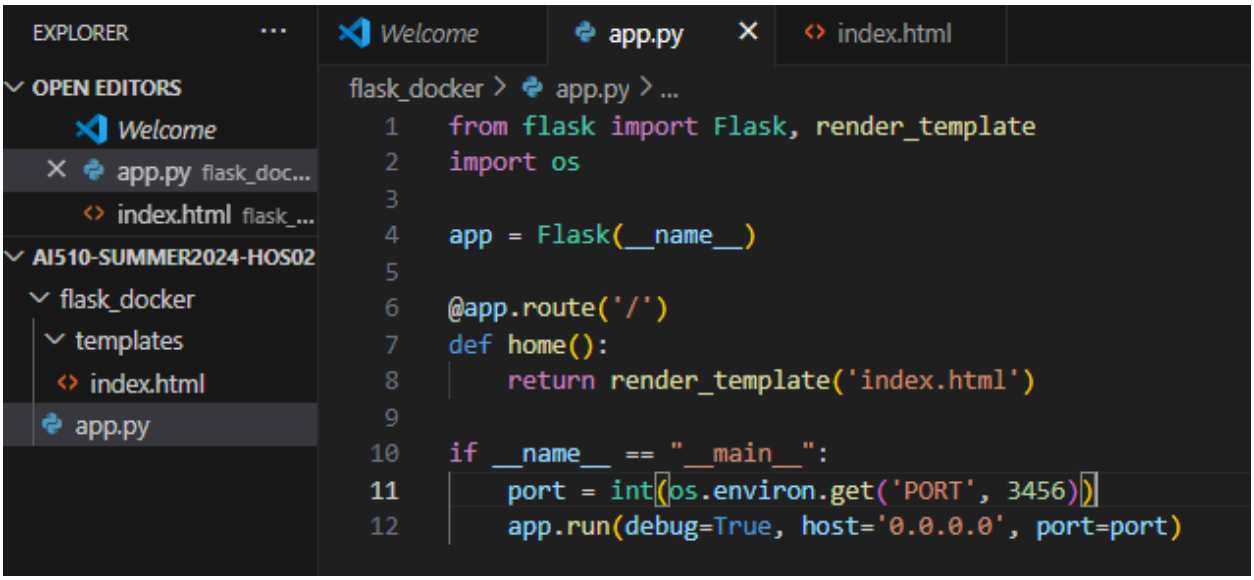

```
PS C:\Users\Administrator\Desktop\TA\Summer\AI510\AI510-Summer2024-HOS02> mkdir flask_docker


    Directory: C:\Users\Administrator\Desktop\TA\Summer\AI510\AI510-Summer2024-HOS02


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         6/10/2024   4:42 PM                flask_docker


PS C:\Users\Administrator\Desktop\TA\Summer\AI510\AI510-Summer2024-HOS02> cd flask_docker
```

Step 2) Go to "flask_docker" folder (ex. Use cd command) and type the following code to install Flask.

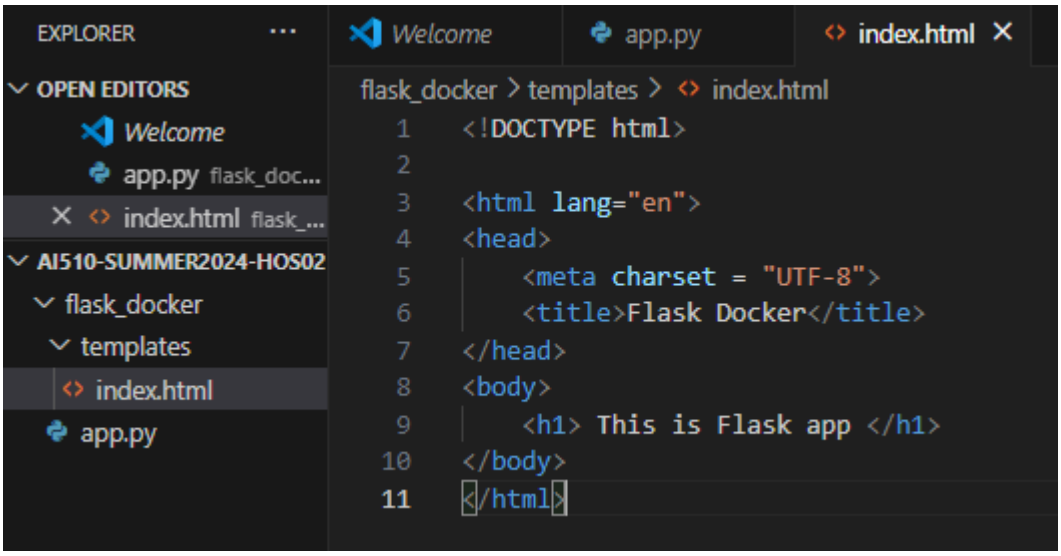
```
PS C:\Users\Administrator\Desktop\TA\Summer\AI510\AI510-Summer2024-HOS02> cd flask_docker
PS C:\Users\Administrator\Desktop\TA\Summer\AI510\AI510-Summer2024-HOS02\flask_docker> pip install Flask
Collecting Flask
  Using cached flask-3.0.3-py3-none-any.whl.metadata (3.2 kB)
Collecting Werkzeug>=3.0.0 (from Flask)
  Using cached werkzeug-3.0.3-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\administrator\appdata\local\programs\python\python312\lib\site-packages (from Flas
)
Collecting itsdangerous>=2.1.2 (from Flask)
  Using cached itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click>=8.1.3 (from Flask)
  Using cached click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from Flask)
```

Note: This is like your PE01 and HOS01. If you are reusing the previous environment, we are ensuring that the Flask framework is installed.

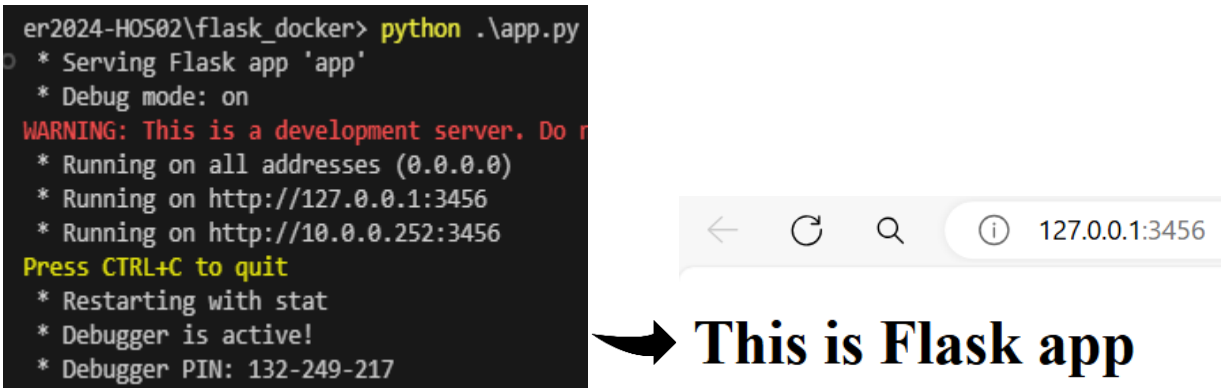
Step 3) Similar to HOS01A/PE01, create a "app.py" python file in the "flask_docker" folder which receives and responds to requests to Flask application. Type the following in the "app.py" file.



Step 4) We will take one extra step from HOS01A/PE01, and this time we will have a front-end page. Create a “templates” folder and then create the “index.html” file. Type the following in the “index.html” file.

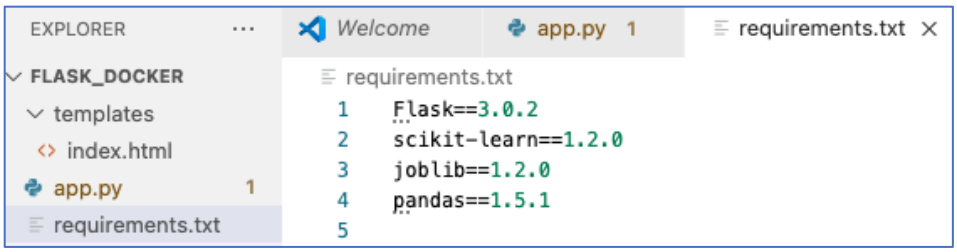


Step 5) Let’s do a host OS(OSX/Windows) flask app test. Type “python app.py” command in the VSCode terminal panel. When the server starts, open web browser to see if the page is shown by opening host address(ex. http://127.0.0.1:3456/). After checking web page opens, press CTRL+C to quit Flask application in VSCode terminal panel.



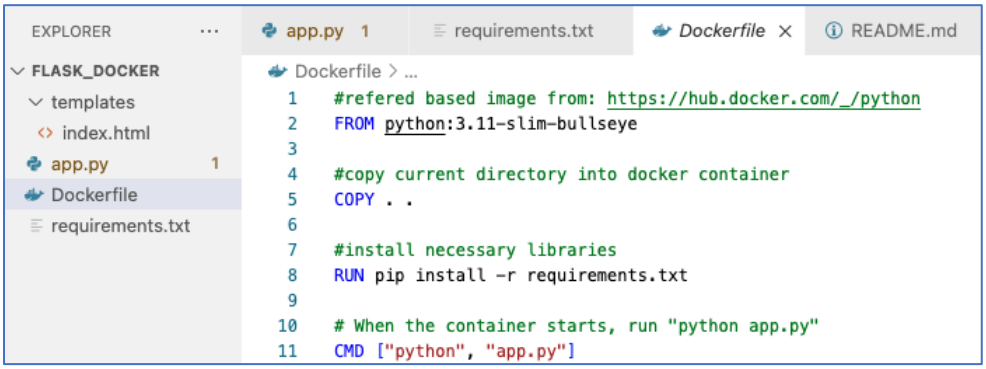
Note: Our “app.py” root path (‘/’) returns index page and web browser page shows what we created in the “index.html” file.

Step 6) The Docker container is a whole new OS environment with minimal installations. We will need all the packages needed to make the Flask service run. In the “flask_docker” folder, create a “requirements.txt” file and type the following in the file.



Note: We just need the Flask package to have the Flask application. However, other libraries are listed to train and serve the model.

Step 7) Dockerfile is an instruction set to build a docker container image. We will define which docker base image to use, how contents should be copied into the container, and how the service should start in our Dockerfile. In the "flask_docker" folder, create a "Dockerfile" file and type the following in the file.



Note: After the image is built, the container image uses baked "ENTRYPOINT" or "CMD" to start the service.

Step 8) Run following commands in terminal to build docker image. Ensure to use "." at the end of command. The "." character means where the build directoy starts from which helps docker build command locate "Dockerfile". The "-t" specifies the name of currently built image.

Command:

```
docker image build -t flask_docker_image .
```

```
mer2024-HOS02\flask_docker> docker image build -t flask_doc
ker_image .
[+] Building 82.7s (8/8) FINISHED          docker:default
 => [internal] load build definition from Dockerfile  0.4s
 => => transferring dockerfile: 146B                  0.0s
 => [internal] load metadata for docker.io/library/p  3.4s
 => [internal] load .dockerignore                     0.3s
 => => transferring context: 2B                       0.0s
 => [internal] load build context                     0.4s
 => => transferring context: 431B                     0.0s
 => [1/3] FROM docker.io/library/python:3.11-slim-b  13.1s
 => => resolve docker.io/library/python:3.11-slim-bu  0.6s
 => => sha256:a24e509d2f7055675a3961 1.65kB / 1.65kB  0.0s
```

Step 9) Let's start just built docker image with following command. When the server starts, open the web browser to see if the page is shown by opening the host address(ex. http://127.0.0.1:3456).

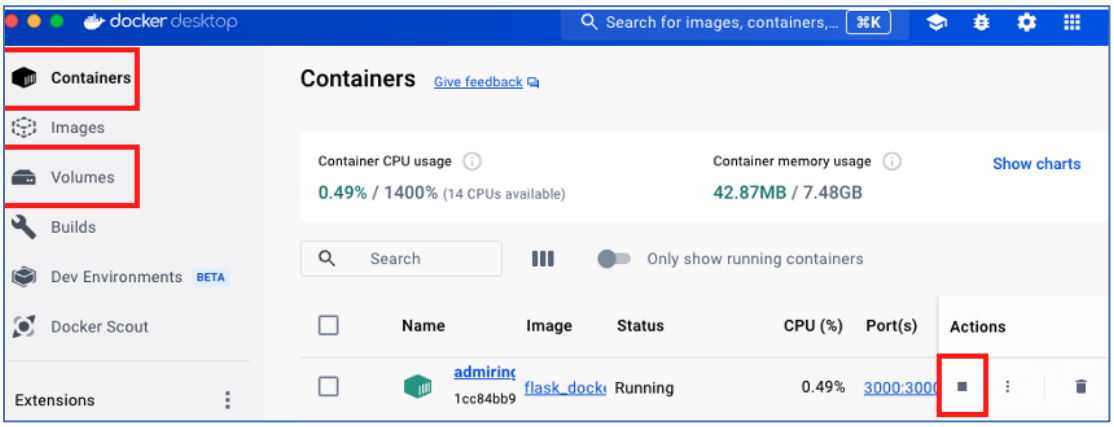
Command:

```
docker run --rm -p 3456:3456 flask_docker_image
```

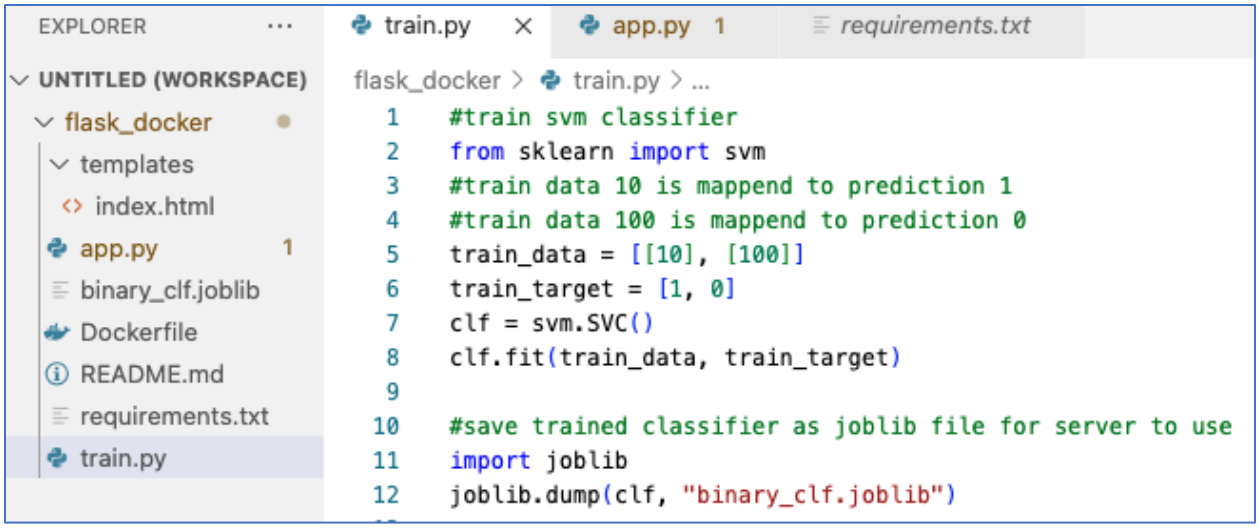
1) "`--rm`" means commanding Docker Daemon to clean up the container and remove the file system after the container exits.
2) "`-p`" means port mapping between the external port to the container's internal port when the container starts
3) "`flask_docker_image`" the last part was the image name we just built from step 8

Note: Press CTRL+C to quit the Flask application in the VSCode terminal panel. We also can pass "`-d`" flag to start the docker service without showing the console output as well as have the service running even when we close the terminal. For example, "`docker run -d --rm -p 3456:3456 flask_docker_image`"

Step 10) Stop the container by opening the "Docker Desktop" application, in the "Containers" tab, and click the "stop" button.
Note: You can free up your disk space by visiting the "Volumes" tab, and clicking the "delete" button on each of the built test images.



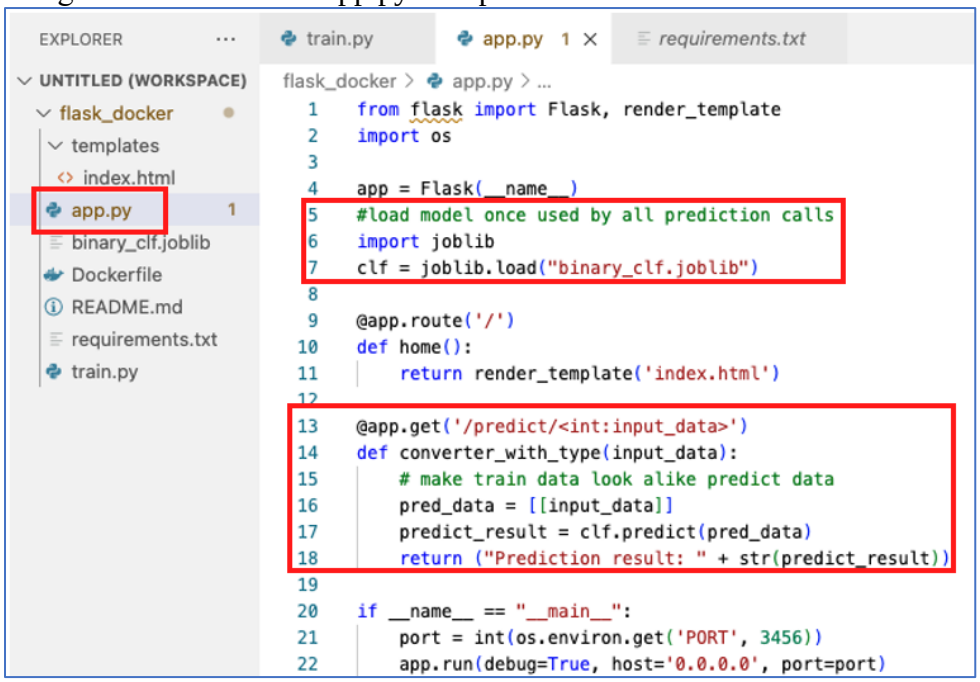
Step 11) In MLOps container images are used to host models. We will create "train.py" script to first train model in the "flask_docker" folder. Create "train.py" file and enter following contents to the file.



Step 12) Run training with "python train.py" command in terminal.

Note: trained classifier object is stored as "binary_clf.joblib"

Step 13) We will do a small addition to the "app.py" script to host trained model using Flask. Add necessary changes as below to the "app.py" script.



Step 14) Let's rerun image build and start container to test predict path with test data through URL.

1) **Build** image(include "**.**" char):

```
docker image build -t flask_docker_image .
```

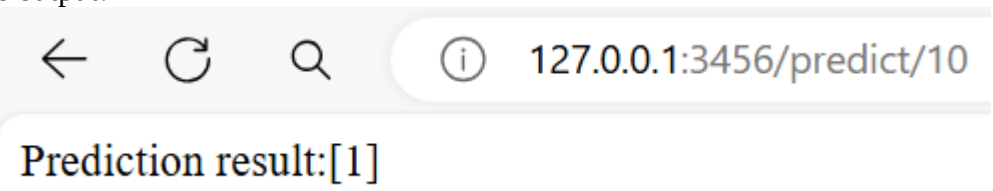
2) **Run** the container:

```
docker run --rm -p 3456:3456 flask_docker_image
```

3) **Open** server URL with **predict** and **data**: http://127.0.0.1:3456/**predict/10**

4) **Change** server URL with **predict** and **data:** : http://127.0.0.1:3456/**predict/100**

Example output:



HOS submission instructions:

1. Please install the GitHub Desktop: https://cityuseattle.github.io/docs/git/github_desktop/
2. Clone, organize, and submit your work through GitHub Desktop: https://cityuseattle.github.io/docs/hoporhos