

SEASON 5

SUPER AI ENGINEER

AI INNOVATOR • AI ENGINEER • AI RESEARCHER



Fast API and Docker

Sivaphong Niyomphanich

SuperAI Season#5



About Me

- Sivaphong Niyomphanich
- Pong+
- Hospital Management Services
- PRINC Group
- SuperAI Engineer SS#2

SuperAI Engineer Season#5



Agenda

- Fast API
- Docker
- Related Fundamental

Pre-Require Tools

- VSCode
- Postman
- Python (Mini-conda and conda-forge)
- WSL
- Docker
- Ngrok
- Please refer document “FastAPI-Docker_ToolsInstall-Manual”

About Docker License Concern

- Docker Engine
 - Option I: Windows OS
 - WSL with Ubuntu and Docker (CE)
 - Or Docker Desktop (License concern*)
 - Option II: Mac OS
 - Docker Desktop (License concern*)
 - Option III: Linux/Unix
 - Docker (CE)

*License concern: Docker Desktop is free for small businesses (fewer than 250 employees AND less than \$10 million in annual revenue), personal use, education, and non-commercial open source projects. Governance also purchase the license.

About Anaconda License Concern

- Anaconda/Conda
 - Option I: Anaconda default repository channel
 - License concern**
 - Option II: Anaconda with conda-forge repository channel
 - The conda-forge channel provides a vast array of packages and is completely open-source, which helps in avoiding any potential licensing issues.
 - Option III: Using other Python
 - Using other package management and
 - environment management.

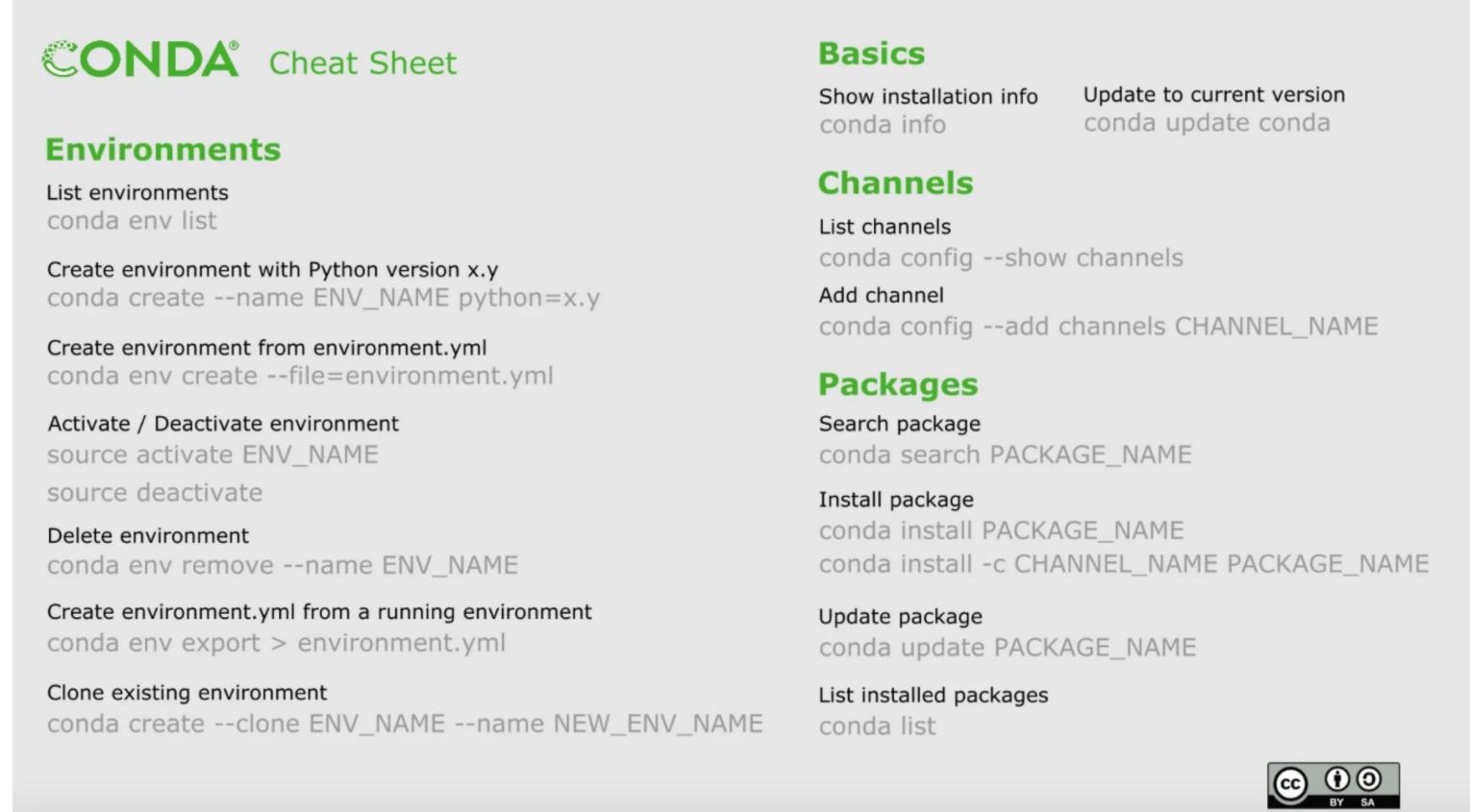
**Anaconda's licensing policies have made some organizations with more than 200 employees, look for alternatives. Who wants to pay 600USD for one license.



Python and Conda

Conda command

- Conda is Python package management and environment management.



The image shows a "CONDA® Cheat Sheet" with sections for Environments, Basics, Channels, and Packages. It includes various Conda commands and their descriptions.

Section	Description	Command
Environments	Basics	Show installation info conda info
	Channels	Update to current version conda update conda
	Packages	Add channel conda config --add channels CHANNEL_NAME
	Basics	List channels conda config --show channels
	Channels	Search package conda search PACKAGE_NAME
	Packages	Install package conda install PACKAGE_NAME conda install -c CHANNEL_NAME PACKAGE_NAME
	Basics	Activate / Deactivate environment source activate ENV_NAME source deactivate
	Channels	Update package conda update PACKAGE_NAME
	Packages	Delete environment conda env remove --name ENV_NAME
Basics	Create environment.yml from a running environment conda env export > environment.yml	
Channels	List installed packages conda list	
Packages	Clone existing environment conda create --clone ENV_NAME --name NEW_ENV_NAME	

Image source: <https://dcordero.medium.com/conda-cheat-sheet-e95ca5aeda85>

Preparing Python for development

- Activate Conda
 - source ~/miniconda3/bin/activate
- List conda environment
 - conda env list

```
pongier@Windows11:~$ source ~/miniconda3/bin/activate
(base) pongier@Windows11:~$ conda env list

# conda environments:
#
base          * /home/pongier/miniconda3
```

- List Python version
 - conda search python

```
(base) pongier@Windows11:~$ conda search python
Loading channels: done
# Name          Version      Build Channel
python          1.0.1        0  conda-forge
python          1.2          0  conda-forge
python          1.3          0  conda-forge
python          1.4          0  conda-forge
python          1.5.2        0  conda-forge
python          1.6          0  conda-forge
python          2.0          0  conda-forge
python          2.6.9        0  conda-forge
python          2.7.12       0  conda-forge
python          2.7.12       1  conda-forge
python          2.7.12       2  conda-forge
python          2.7.13       0  conda-forge
python          2.7.13       1  conda-forge
```

Preparing Python for development

- Create conda environment
 - `conda create --name superapi python=3.12.9`

```
(base) pongier@Windows11:~$ conda create --name superapi python=3.12.9
Channels:
- conda-forge
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done
```

- Deactivate conda environment
 - `conda deactivate`

```
(base) pongier@Windows11:~$ conda deactivate
pongier@Windows11:~$
```

Preparing Python for development

- Activate superapi environment
 - `source ~/miniconda3/bin/activate superapi`

```
pongier@Windows11:~$ source ~/miniconda3/bin/activate superapi
(superapi) pongier@Windows11:~$ |
```

- Install package/library
 - `conda install numpy`
 - Or `pip install numpy`

```
(superapi) pongier@Windows11:~$ conda install numpy
Channels:
 - conda-forge
 - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done
```

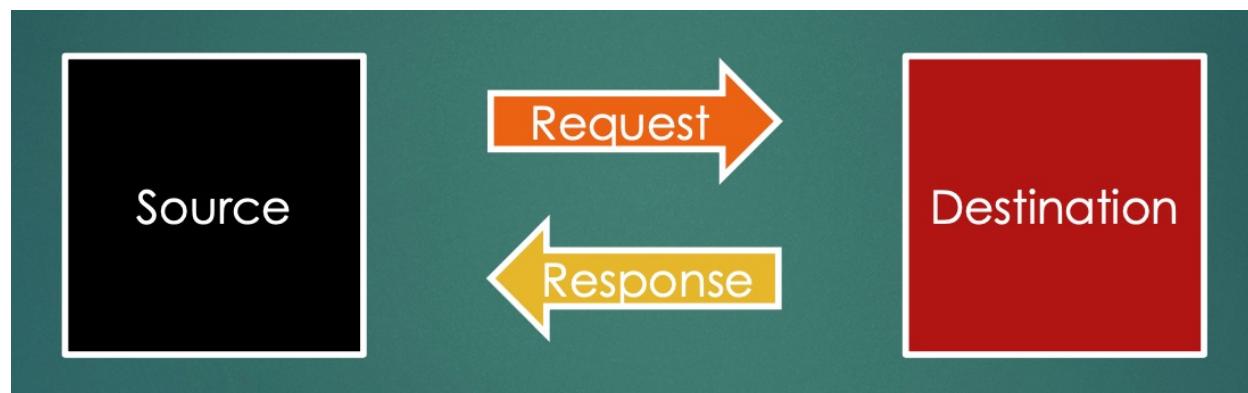
#5



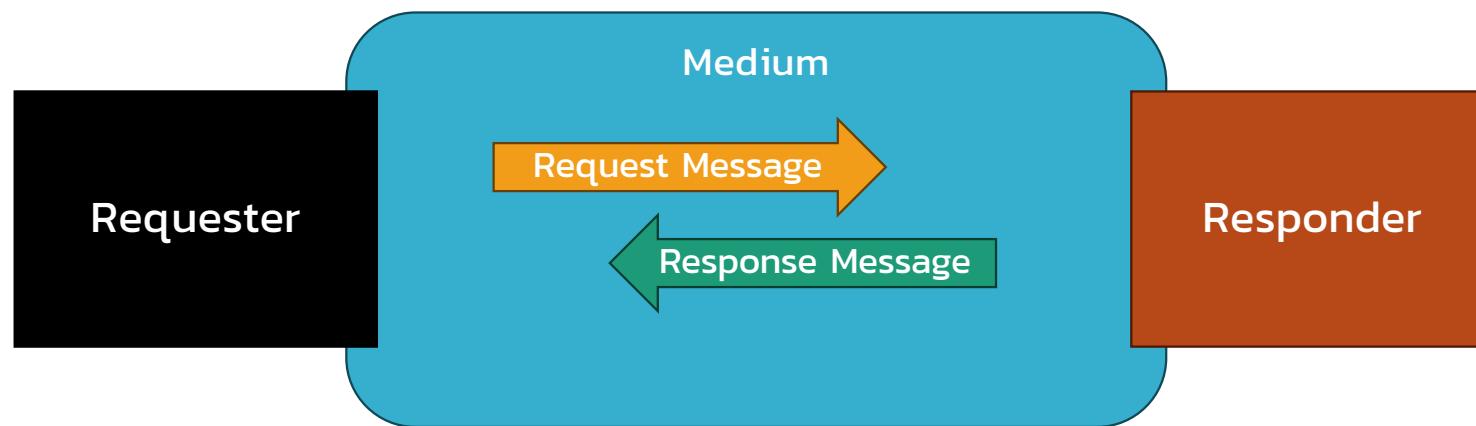
Fast API

API (Application Programming Interface) Overview

- APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols.



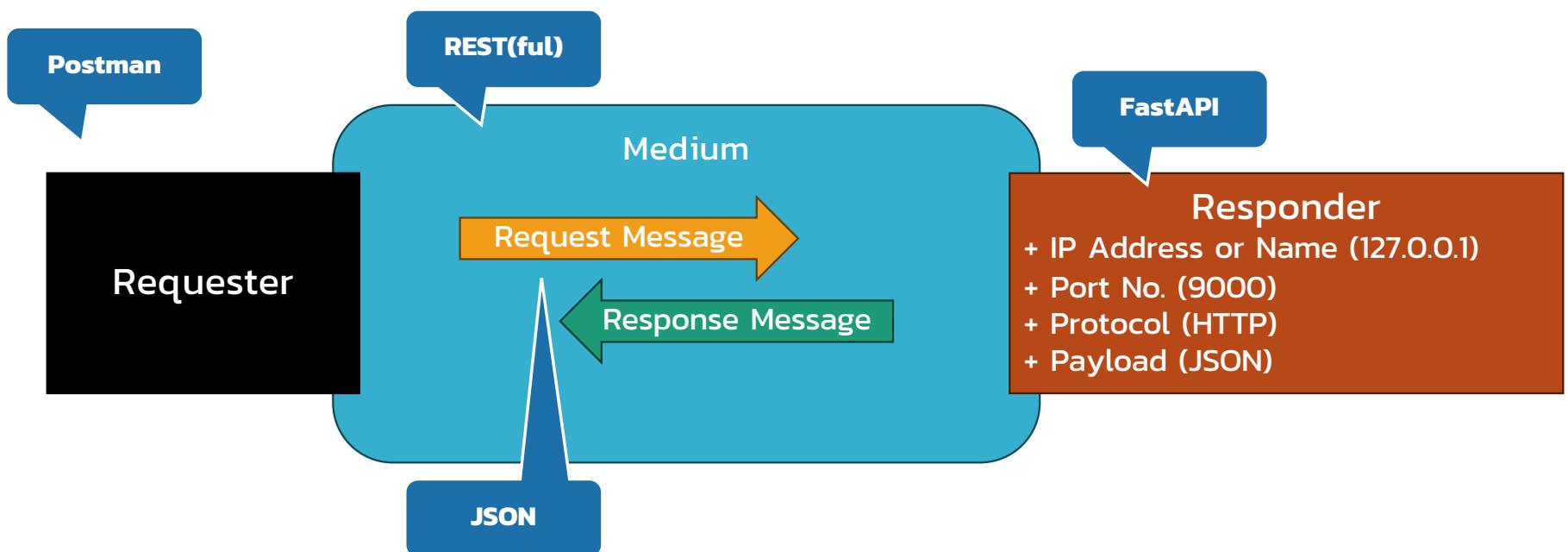
API Architecture



API Architecture

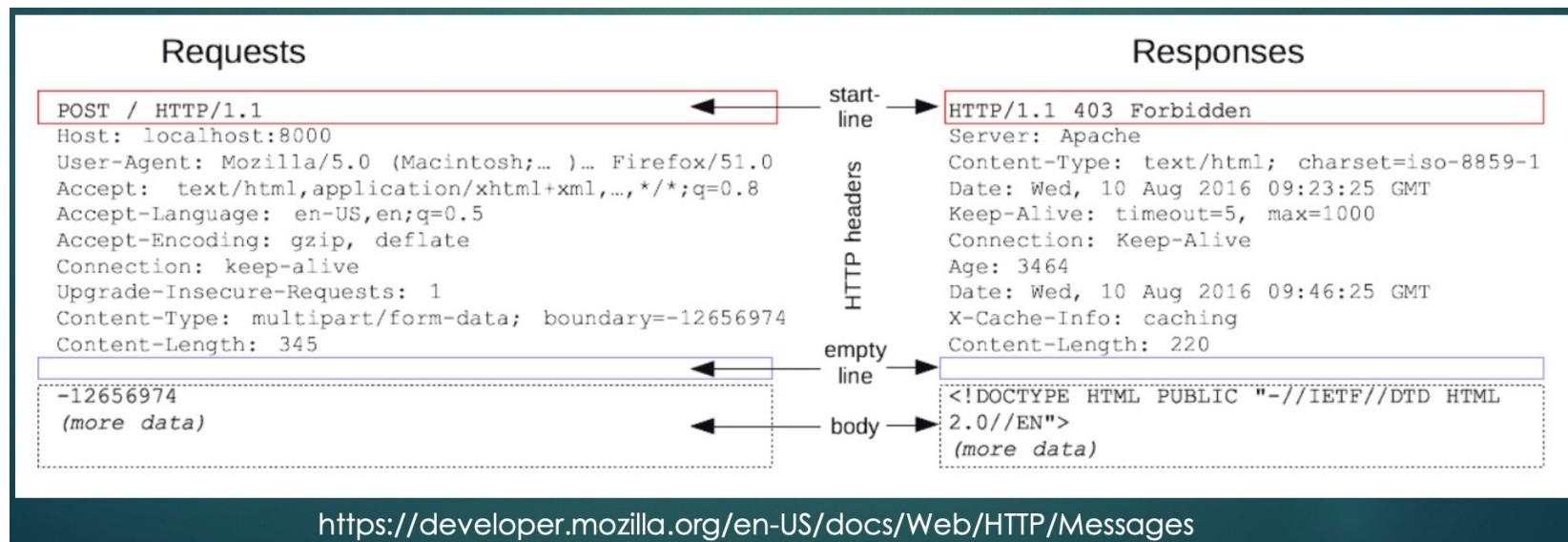
- **Requester**
 - Initial caller
- **Message**
 - JSON
 - XML
- **Medium**
 - SOAP
 - RPC
 - Web Socket
 - REST
- **Responder**
 - Response result

API Architecture for today



Understanding HTTP Protocol

- HTTP Header
- HTTP Body



Understanding HTTP URL

HTTP URL Anatomy

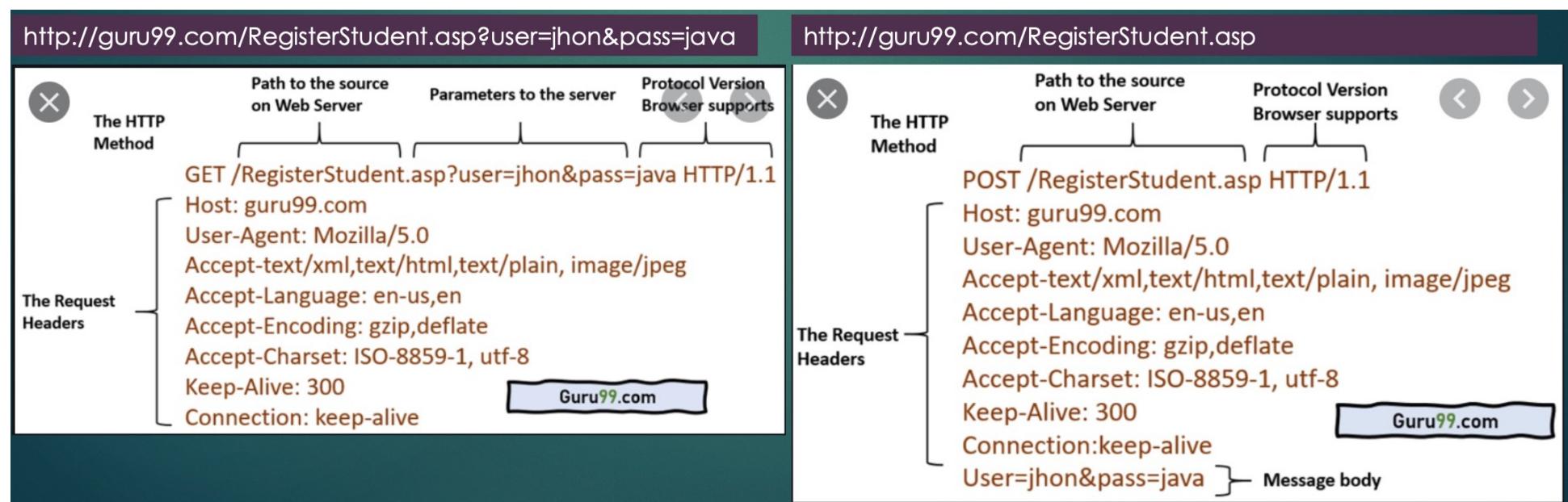
https://www.example.com:3000/path/resource?id=123#section-id

Key

- 1 Scheme - defines how the resource will be obtained.
- 2 Subdomain - www is most common but not required.
- 3 Domain - unique value within its top-level domain.
- 4 Top-level Domain - hundreds of options now exist.
- 5 Port - if omitted HTTP will connect on port 80, HTTPS on 443.
- 6 Path - specify and perhaps find requested resource.
- 7 Query String - data passed to server-side software, if present.
- 8 Fragment Identifier - a specific place within an HTML document.

Understanding HTTP Request Method

- GET Method
- POST Method



- Also PUT, PATCH, DELETE

Important when working with (REST) API

- Request URL (IP_Name/Port No./Path)
- HTTP Request Method
- Parameter/Payload
 - Request/Response
- Is need subscriber (e.g. token, key)

Call REST API with Postman

- Simple API
- <https://agify.io/documentation>
- Request URL (IP_Name/Port No./Path)
 - `https://api.agify.io?name=michael`
- HTTP Request Method
 - GET
- Parameter/Payload
 - `name=michael`
- Request/Response
 - `{ "count": 298219, "name": "michael", "age": 62 }`
- Is need subscriber (e.g. token, key)
 - No

Call REST API with Postman

The screenshot shows the Postman application interface. At the top, there is a search bar with the URL `GET https://api.agify.io?name=pong`. Below the search bar, the request details are shown: method `GET`, URL `https://api.agify.io?name=pong`, and a `Send` button. Under the `Params` tab, there is a table for `Query Params` with one entry: `name` with value `pong`. The `Body` tab is selected, showing the JSON response:

```
1 {  
2   "count": 1246,  
3   "name": "pong",  
4   "age": 60  
5 }
```

At the top right of the main area, it says `Status: 200 OK Time: 1557 ms Size: 604 B` and there is a `Save Response` button. Below the JSON response, there are tabs for `Pretty`, `Raw`, `Preview`, and `Visualize`, along with a `JSON` dropdown and a search icon.

Call REST API with Postman

- More complex API
- <https://reqbin.com/req/alaltavu/sample-api-post-request>
- Request URL (IP_Name/Port No./Path)
 - <https://reqbin.com/echo/post/json>
- HTTP Request Method
 - POST
- Parameter/Payload
- Request/Response
- Is need subscriber (e.g. token, key)
 - No

```
{  
  "Id": 78912,  
  "Customer": "Jason Sweet",  
  "Quantity": 1,  
  "Price": 18.00  
}  
  
{  
  "success": "true"  
}
```

Call REST API with Postman

The screenshot shows the Postman application interface. At the top, there are two tabs: 'GET https://api.agify.io?name=' and 'POST https://reqbin.com/echo'. Below these tabs is a search bar containing 'https://reqbin.com/echo/post/json' and a 'Save' button.

The main area shows a 'POST' method selected for the URL 'https://reqbin.com/echo/post/json'. To the right of the URL is a 'Send' button. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is active, showing the following JSON payload:

```
1 {  
2   "Id": 78912,  
3   "Customer": "Jason Sweet",  
4   "Quantity": 1,  
5   "Price": 18.00  
6 }
```

Below the body editor, there are tabs for 'Body', 'Cookies', 'Headers (16)', and 'Test Results'. The 'Body' tab is active. On the right side of the interface, there is a status bar showing 'Status: 200 OK', 'Time: 393 ms', 'Size: 959 B', and a 'Save Response' button. At the bottom, there are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON' (with a dropdown arrow), along with a search icon.



Create our API with FastAPI

- FastAPI is a modern, fast (high-performance), web framework for building APIs with Python based on standard Python type hints.
- <https://fastapi.tiangolo.com>

Preapring for Coding

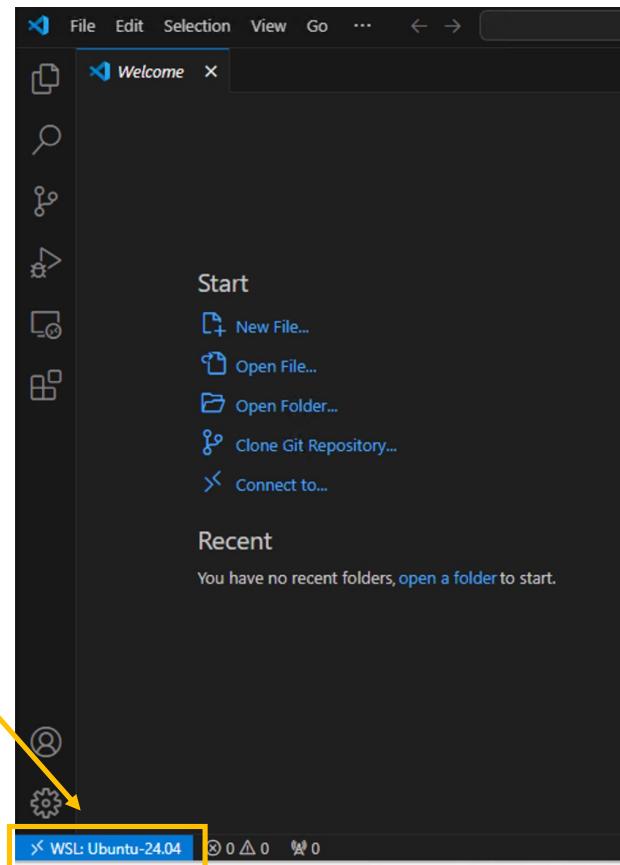
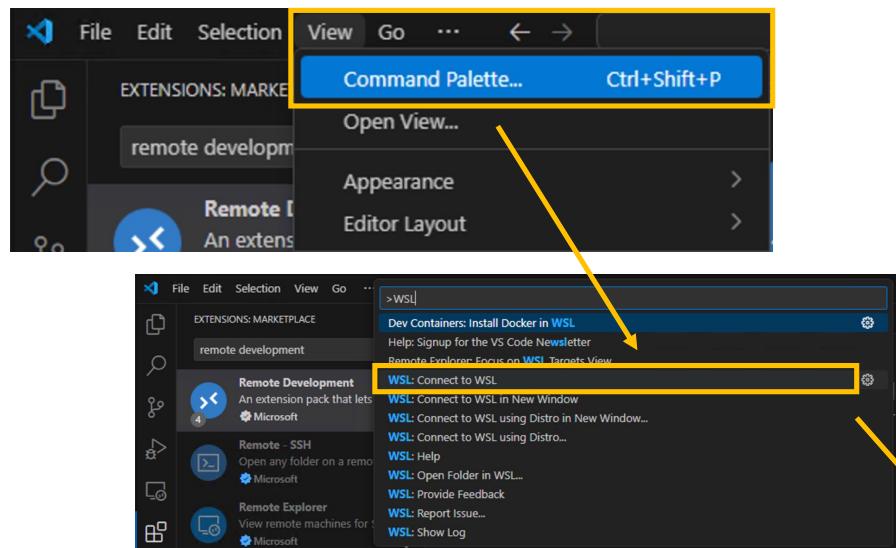
- What is your computer?
 - Windows user may using WSL with Ubuntu and Mini-conda
 - Mac user may using MacOSX and Mini-conda
 - Linux user may using Linux OS and Mini-conda
- Coding at
 - VSCode
- Testing tool by
 - Postman

Preapring for Coding

- Python and Environment
 - **Using python environment named superapi**
 - Using package/library
 - "fastapi[standard]"

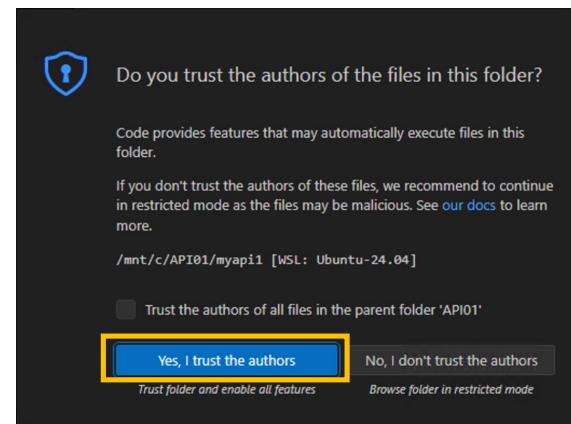
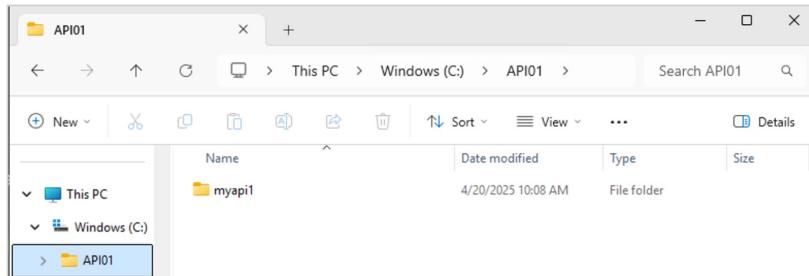
Open VSCode and connect to WSL

- Open VSCode
- Connect to WSL

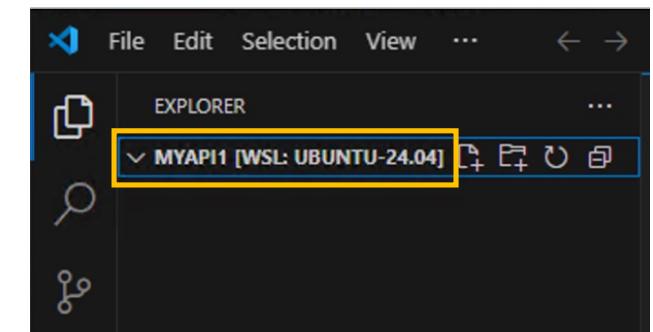
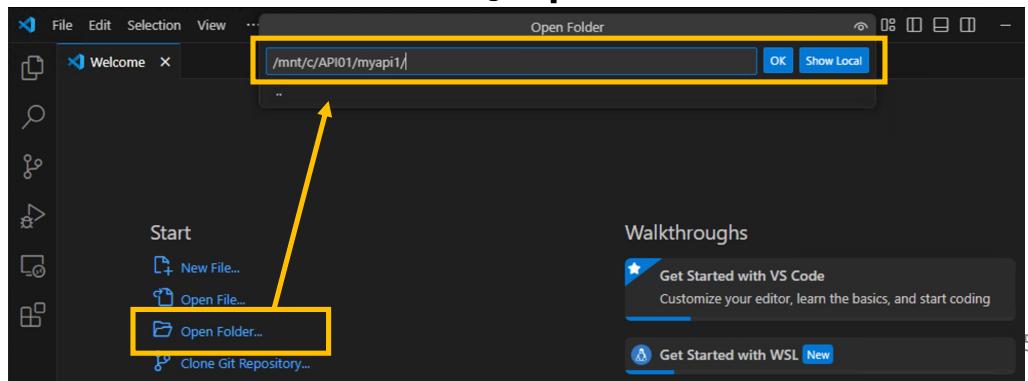


Create API01\myapi1 folder on C:\

- Create API01\myapi1 folder on C:\ at Windows OS

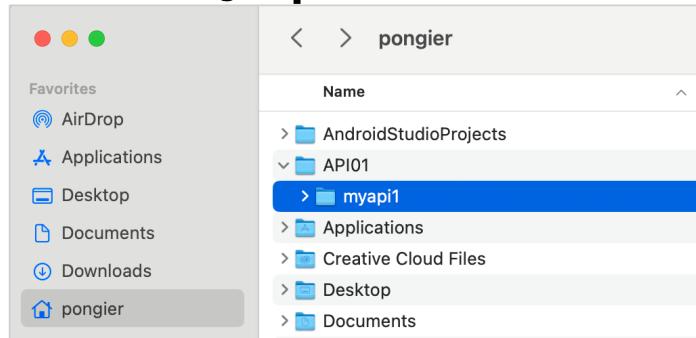


- In VSCode -> “Open Folder”
 - /mnt/c/API01/myapi1/

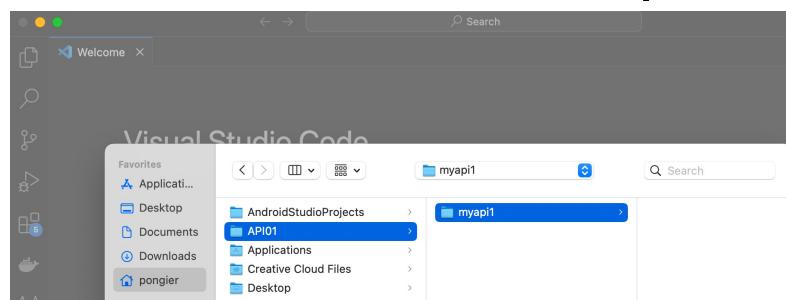


Open VSCode and Create directory on

- Create API01\myapi1 folder on /Users/YOUR_NAME at MacOSX

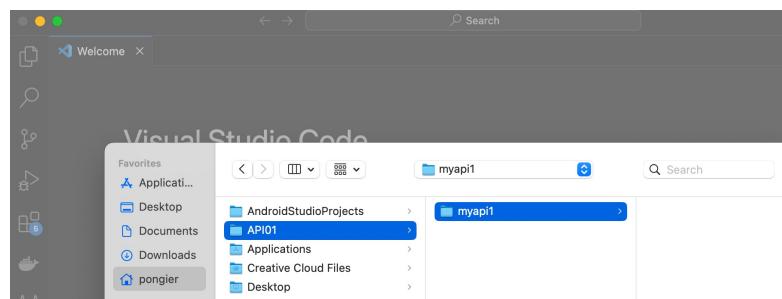


- In VSCode -> "Open"
 - /Users/YOUR_NAME/API01/myapi1/



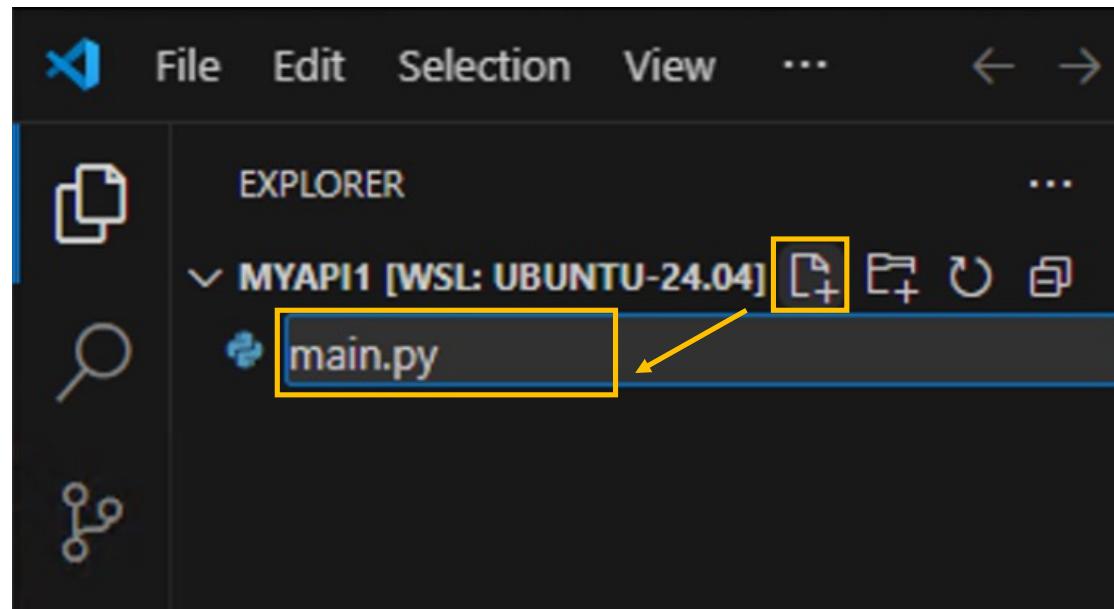
Open VSCode and Create directory on

- Create API01\myapi1 folder on /home/YOUR_NAME at Linux
- In VSCode -> “Open”
 - /home/YOUR_NAME/API01/myapi1/



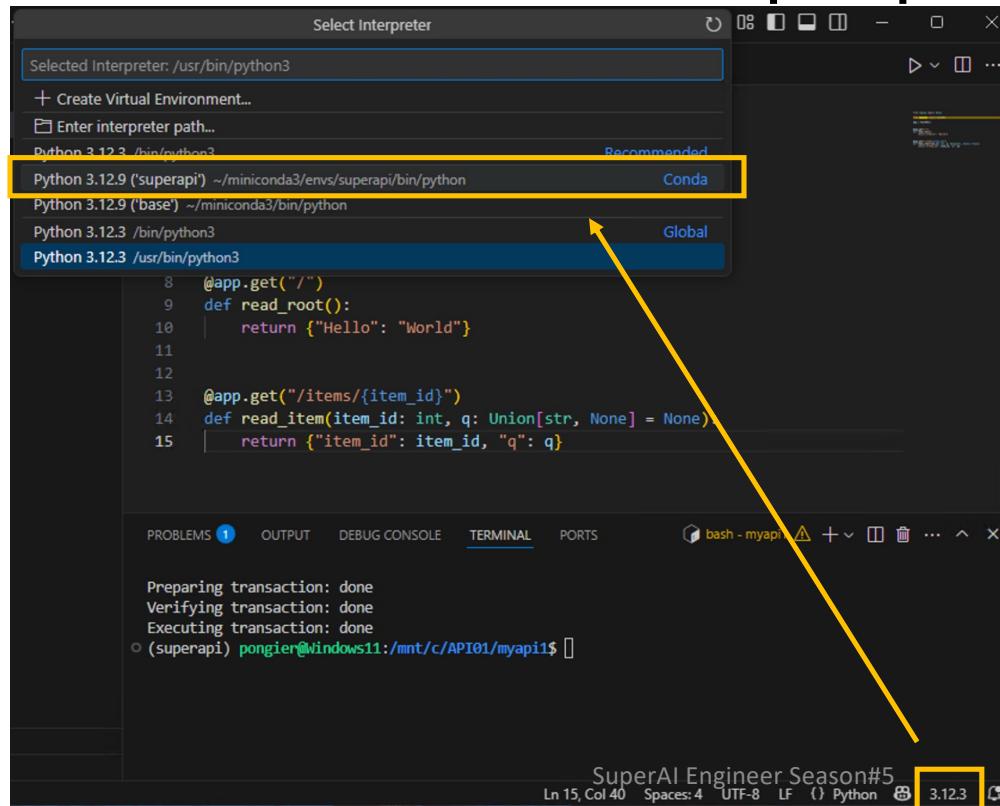
Create a new file named main.py

- Create a new file named “main.py” in VSCode



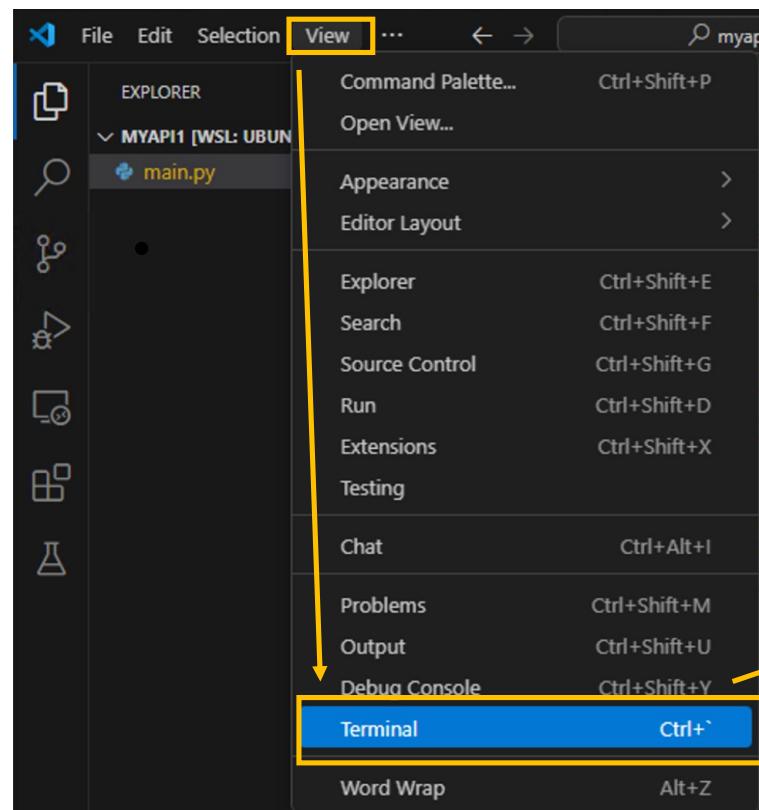
Select Python/Conda environment

- Select environment named “superapi” in VSCode



Install fastapi package/library

- Install package/library named fastapi[standard] in VSCode terminal



conda install "fastapi[standard]"

A screenshot of the VSCode terminal window. The command 'conda install "fastapi[standard]"' is entered and highlighted with a yellow box. The output shows the process of installing the package, including channel information, platform details, and a warning about a newer version of conda. The status bar at the bottom shows 'Ln 15, Col 40' and other terminal settings.

```
(superapi) pongier@Windows11:/mnt/c/API01/myapi1$ conda install "fastapi[standard]"
Channels:
- conda-forge
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

=> WARNING: A newer version of conda exists. <==
      current version: 25.1.1
```

Coding in file main.py

```
from typing import Union

from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"Hello": "World"}

@app.get("/items/{item_id}")
def read_item(item_id: int, q: Union[str, None] = None):
    return {"item_id": item_id, "q": q}
```

Running your code

- Enter command in VSCode terminal
 - `fastapi dev main.py`

The screenshot shows the VSCode interface with the following details:

- File Explorer:** Shows a single file named `main.py`.
- Terminal:** The command `fastapi dev main.py` is being run. The output shows the server starting at `http://127.0.0.1:8000` and documentation at `http://127.0.0.1:8000/docs`.
- Status Bar:** Shows the environment as `WSL: Ubuntu-24.04`, line 15, column 40, and Python 3.12.9.

Testing your API

- Testing your API with Postman
 - Request URL (IP_Name/Port No./Path)
 - `http://127.0.0.1:8000`
 - HTTP Request Method
 - GET
 - Parameter/Payload
 - No
 - Request/Response
 - `{ "Hello": "World" }`
 - Is need subscriber (e.g. token, key)
 - No

The screenshot shows the Postman application interface. At the top, there is a header bar with the text "GET http://127.0.0.1:8000" and a "+" button. Below the header, there is a main request configuration area with a "HTTP" icon and the URL "http://127.0.0.1:8000". To the right of the URL is a "Send" button. Below the URL input, there are tabs for "Params", "Auth", "Headers (6)", "Body", "Pre-req.", "Tests", and "Settings". The "Headers" tab is currently selected. Under the "Headers" tab, there is a section for "Query Params" with two rows: "Key" and "Value". In the "Body" section, there is a "Pretty" tab selected, showing the JSON response: `1 "Hello": "World"`. To the right of the body, there is a status bar showing "200 OK 95 ms 142 B" and a "Save Response" button. On the far right, there is a vertical sidebar with icons for file operations.

Testing your API

- Testing your API with Postman
 - Request URL (IP_Name/Port No./Path)
 - `http://127.0.0.1:8000/items/1?q=x`
 - HTTP Request Method
 - GET
 - Parameter/Payload
 - items
 - q
 - Request/Response
 - {
 - "item_id": 1,
 - "q": "sometext"
 - }
 - Is need subscriber (e.g. token, key)
 - No

The screenshot shows the Postman application interface. At the top, there is a header bar with a search field containing 'GET http://127.0.0.1:8000/item'. Below the header, there is a main panel for a GET request to 'http://127.0.0.1:8000/items/1?q=sometext'. The 'Params' tab is selected, showing a table with one row: 'q' (Key) and 'sometext' (Value). The 'Body' tab shows a JSON response with the following content:

```
1 "item_id": 1,  
2 "q": "sometext"
```

Understanding the code (FastAPI)

```
from typing import Union

from fastapi import FastAPI

app = FastAPI()

@app.get("/")      #Method, Path
def read_root():  #Defined function
    return {"Hello": "World"} #Result that return

@app.get("/items/{item_id}") #Method, Path, Parameter/Variable
def read_item(item_id: int, q: Union[str, None] = None): #Defined function
    return {"item_id": item_id, "q": q} #Result that return
```

Understanding JSON

- JSON (Java Script Object Notation)

- Key:Value
- [] is array



The screenshot shows a JSON editor window titled "personal.json". The code is a JSON object with the following structure:

```
1 {  
2     "$schema": "personal-schema.json",  
3     "personnel": [  
4         {"person": [  
5             {"id": "Big.Boss",  
6                 "name": {  
7                     "family": "Boss",  
8                     "given": "Big"  
9                 },  
10                "email": "chief@oxygenxml.com",  
11                "link": {  
12                    "subordinates": [  
13                        "one.worker",  
14                        "two.worker",  
15                        "three.worker",  
16                        "four.worker",  
17                        "five.worker"  
18                    ]  
19                }  
20            ]  
21        ]  
22    ]  
23}
```

More complex your API

```
from typing import Union

from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class Product(BaseModel):
    name: str
    price: float
    id: int

@app.get("/")
def read_root():
    return {"Hello": "World"}

@app.get("/items/{item_id}")
def read_item(item_id: int, q: Union[str, None] = None):
    return {"item_id": item_id, "q": q}

@app.post("/products")
def add_product(product: Product):
    return {"product_name": product.name, "price": product.price*500}
```

Testing your API

- Testing your API with Postman
 - Request URL (IP_Name/Port No./Path)
 - `http://127.0.0.1:8000/products`
 - HTTP Request Method
 - POST
 - Parameter/Payload
 - ```
1 {
2 "name": "Nissan",
3 "price": 20.5,
4 "id": 9
5 }
```
  - Request/Response
    - ```
1 {  
2   "product_name": "Nissan",  
3   "price": 10250.0  
4 }
```
 - Is need subscriber (e.g. token, key)
 - No

The screenshot shows the Postman application interface. At the top, there are two tabs: 'GET http://127.0.0.1:8000/item' and 'POST http://127.0.0.1:8000/pro'. Below these, a search bar contains 'http://127.0.0.1:8000/products'. On the right side of the search bar are 'Save' and 'Send' buttons. Underneath the search bar, there are tabs for 'Params', 'Auth', 'Headers (8)', 'Body' (which is selected), 'Pre-req.', 'Tests', and 'Settings'. To the right of these tabs are 'Cookies' and 'Beautify' buttons. The 'Body' section has 'raw' and 'JSON' dropdowns, with 'JSON' currently selected. Below this is a code editor containing a JSON object:

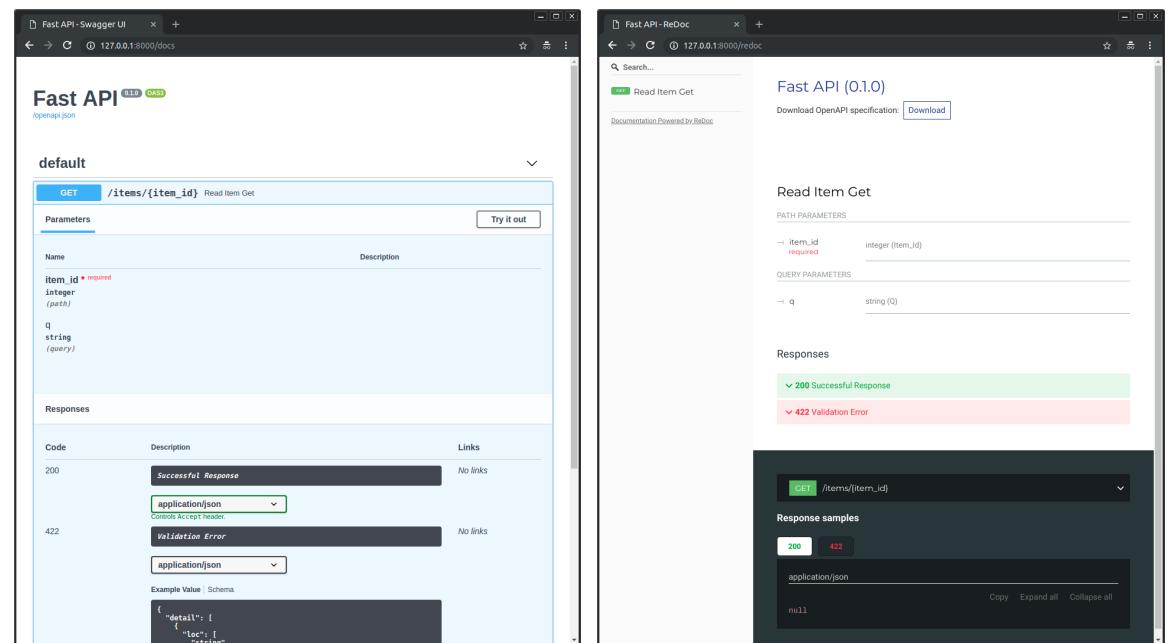
```
1 {  
2   "name": "Nissan",  
3   "price": 20.5,  
4   "id": 9  
5 }
```

At the bottom of the interface, there is a 'Body' dropdown set to 'Pretty', and a status bar showing '200 OK 20 ms 166 B' and a 'Save Response' button. Below the status bar is another code editor showing the same JSON object in pretty-printed format:

```
1 {  
2   "product_name": "Nissan",  
3   "price": 10250.0  
4 }
```

Something fantastic

- <http://127.0.0.1:8000/docs>
- <http://127.0.0.1:8000/redoc>



Custom Port and Running in Production

- Using uvicron
 - Bind all IP and Port is 9000
 - Multi worker is 2
- Install package/library
 - conda install "uvicorn[standard]"
- Running API
 - uvicorn main:app --host 0.0.0.0 --port 9000 --workers 2

```
(superapi) pongier@Windows11:/mnt/c/API01/myapi1$ conda install "uvicorn[standard]"
Retrieving notices: done
Channels:
- conda-forge
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done
```

```
(superapi) pongier@Windows11:/mnt/c/API01/myapi1$ uvicorn main:app --host 0.0.0.0 --port 9000 --workers 2
INFO:     Uvicorn running on http://0.0.0.0:9000 (Press CTRL+C to quit)
INFO:     Started parent process [6290]
INFO:     Started server process [6292]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Started server process [6293]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Entenable complexities your API

- FastAPI stands on the shoulders of giants:
 - Starlette for the web parts.
 - Pydantic for the data parts.
- Additional optional Pydantic dependencies:
 - pydantic-settings – for settings management.
- Used by FastAPI / Starlette:
 - uvicorn – for the server that loads and serves your application. This includes
 - uvicorn[standard], which includes some dependencies (e.g. uvloop) needed for high performance serving.

HTTP Method for API

HTTP Verb	CRUD	Entire Collection (e.g. /customers)	Specific Item (e.g. /customers/{id})
POST	Create	201 (Created), 'Location' header with link to /customers/{id} containing new ID.	404 (Not Found), 409 (Conflict) if resource already exists..
GET	Read	200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single customer. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
PATCH	Update/Modify	405 (Method Not Allowed), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
DELETE	Delete	405 (Method Not Allowed), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

Image source: <https://www.restapitutorial.com/lessons/httpmethods.html>

Creating API with your AI Model

- Understand the example AI Model
 - Input are Age, Sex and Embarked
 - Output is Prediction = 0,1
- Install require package/library
 - conda install numpy
 - conda install pandas
 - conda install scikit-learn

Model Code

Source: <https://www.datacamp.com/tutorial/machine-learning-models-api-python>

```
# Import dependencies
import pandas as pd
import numpy as np

# Load the dataset in a dataframe object and include only four features as mentioned
url = "http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
df = pd.read_csv(url)
include = ['Age', 'Sex', 'Embarked', 'Survived'] # Only four features
df_ = df[include]

# Data Preprocessing
categoricals = []
for col, col_type in df_.dtypes.items():
    if col_type == 'O':
        categoricals.append(col)
    else:
        df_[col].fillna(0, inplace=True)

df_ohe = pd.get_dummies(df_, columns=categoricals, dummy_na=True)

# Logistic Regression classifier
from sklearn.linear_model import LogisticRegression
dependent_variable = 'Survived'
x = df_ohe[df_ohe.columns.difference([dependent_variable])]
y = df_ohe[dependent_variable]
lr = LogisticRegression()
lr.fit(x, y)

# Save your model
import joblib
joblib.dump(lr, 'model.pkl')
print("Model dumped!")

# Load the model that you just saved
lr = joblib.load('model.pkl')

# Saving the data columns from training
model_columns = list(x.columns)
joblib.dump(model_columns, 'model_columns.pkl')
print("Models columns dumped!")
```

API Code

```
from typing import Union

from fastapi import FastAPI
from pydantic import BaseModel

import joblib
import pandas as pd
import numpy as np

app = FastAPI()

class Product(BaseModel):
    name: str
    price: float
    id: int

class Idata(BaseModel):
    Age: int
    Sex: str
    Embarked: str

lr = joblib.load("model.pkl") # Load "model.pkl"
model_columns = joblib.load("model_columns.pkl") # Load "model_columns.pkl"

@app.get("/")
def read_root():
    return {"Hello": "World"}

@app.get("/items/{item_id}")
def read_item(item_id: int, q: Union[str, None] = None):
    return {"item_id": item_id, "q": q}

@app.post("/products")
def add_product(product: Product):
    return {"product_name": product.name, "price": product.price*500}

@app.post("/predicts")
def ai_predict(indata: Idata):
    query = pd.get_dummies(pd.DataFrame(data=indata))
    query = query.reindex(columns=model_columns, fill_value=0)
    prediction = list(lr.predict(query))
    return {'prediction': str(prediction)}
```

Testing your API

- Testing your API with Postman

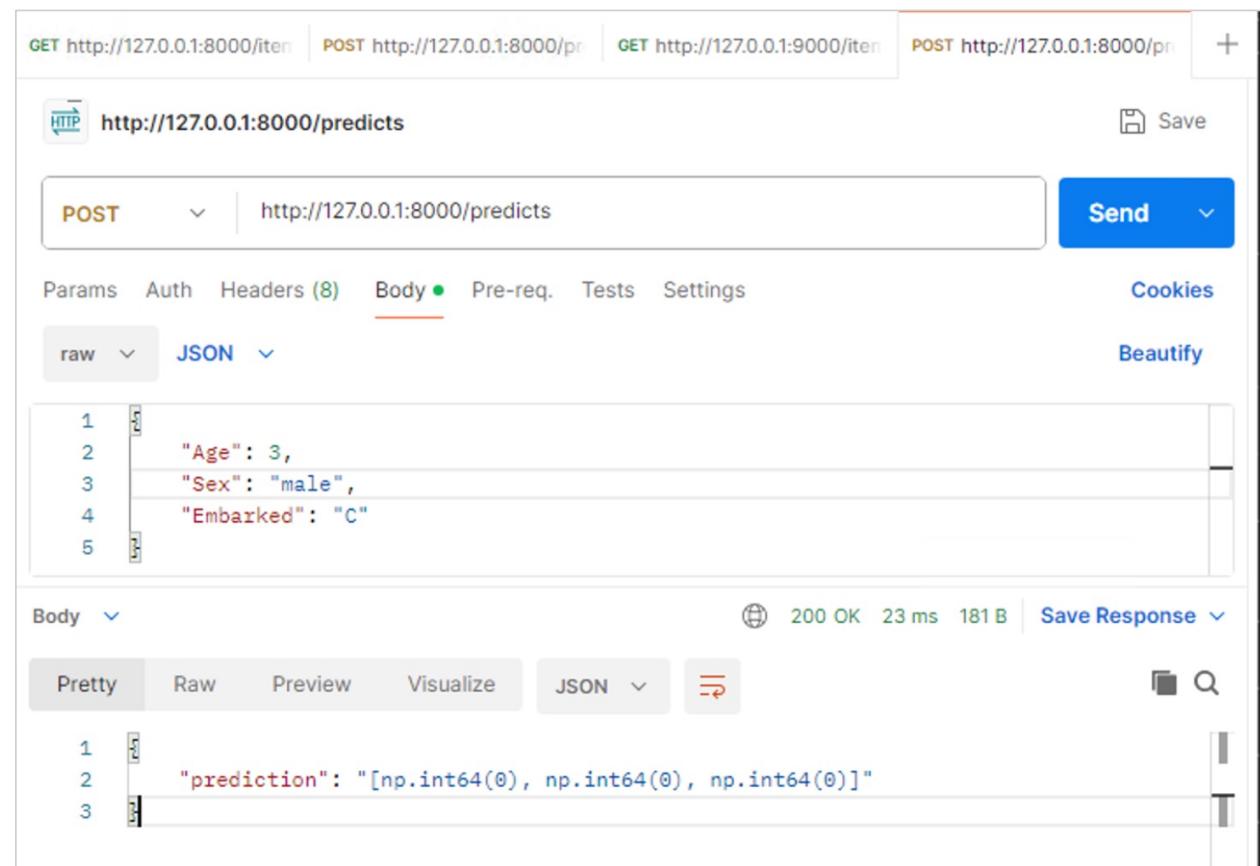
- Request URL (IP_Name/Port No./Path)
 - `http://127.0.0.1:8000/predicts`

- HTTP Request Method
 - POST

- Parameter/Payload
 - {
 - "Age": 24,
 - "Sex": "female",
 - "Embarked": "C"
 - }

- Request/Response
 - {
 - "prediction": "[np.int64(0), np.int64(0), np.int64(0)]"
 - }

- Is need subscriber (e.g. token, key)
 - No



See More...

- FastAPI
 - <https://fastapi.tiangolo.com>
- WSL with VSCode
 - <https://code.visualstudio.com/docs/remote/wsl>
- Miniconda
 - <https://www.anaconda.com/docs/getting-started/miniconda/main>



Docker

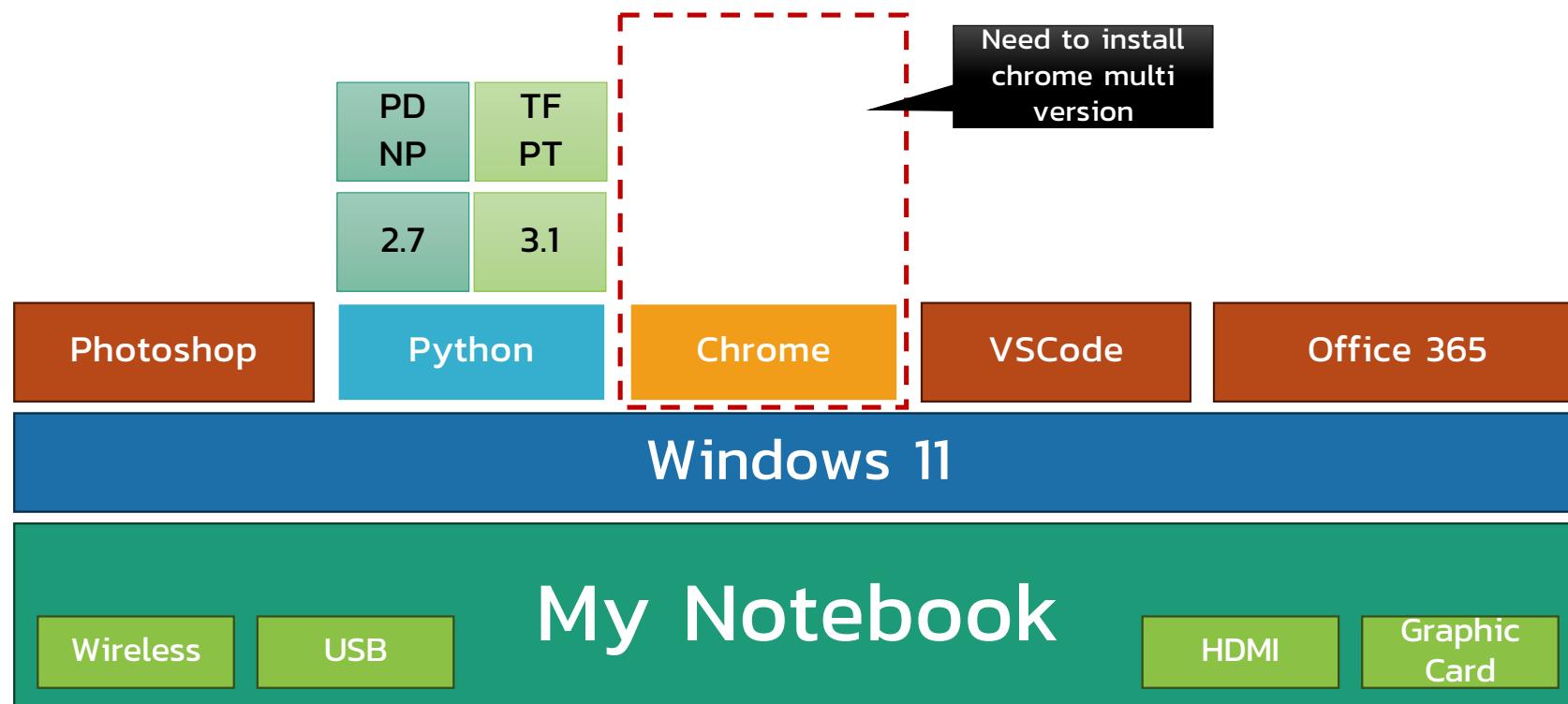
Deploy API in modern world

- Why Docker
 - Running our API on Server or other Computer???
- How Docker
 - Docker usage and command
- Docker Compose
 - Docker and Compose for easier life

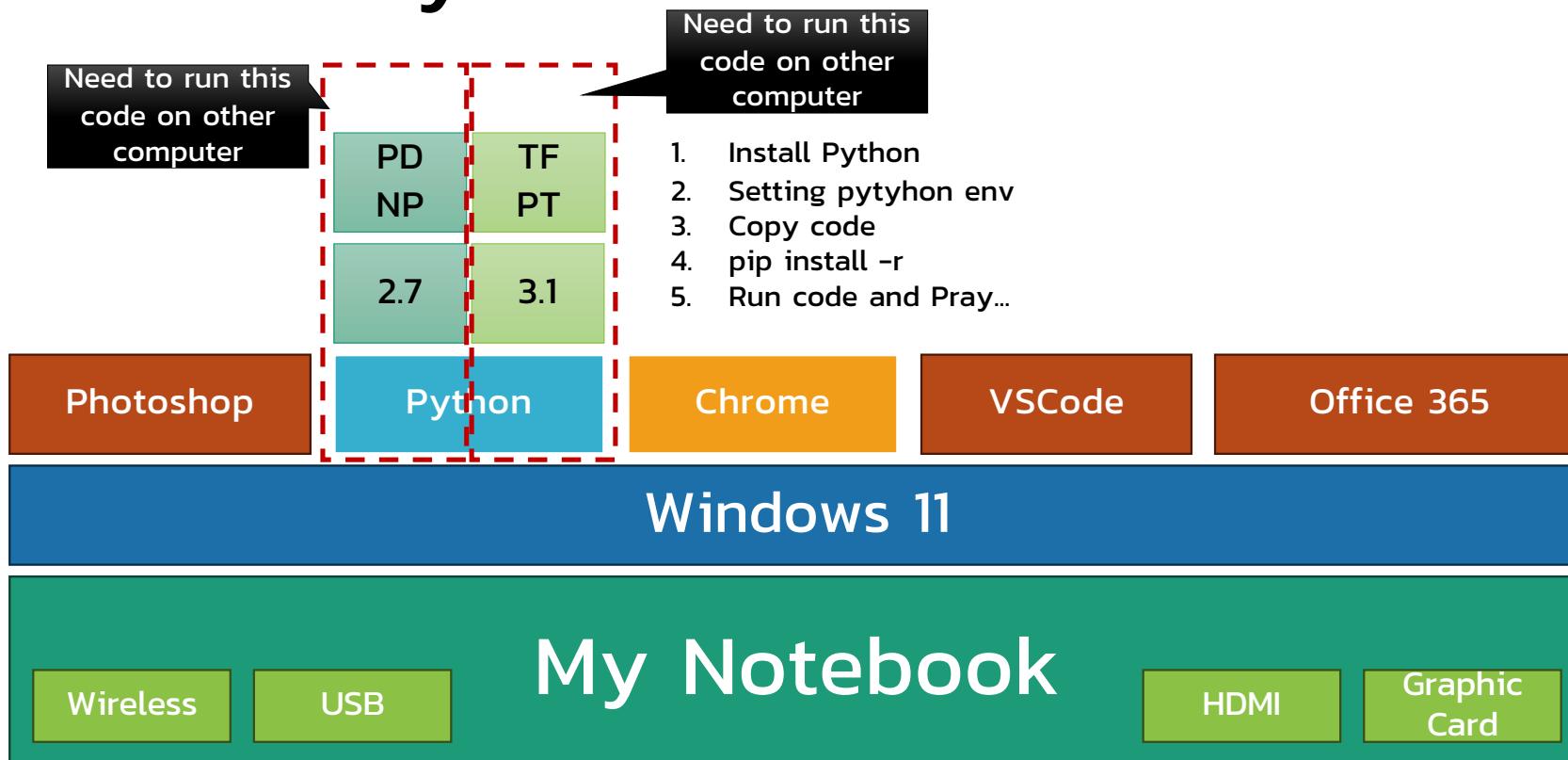
Major Problem on Application Deployment

- Deploy our API or App on Server or Other Computer
- Scaling our API or App for large concurrent of users
- Deployment Strategies and Rollback

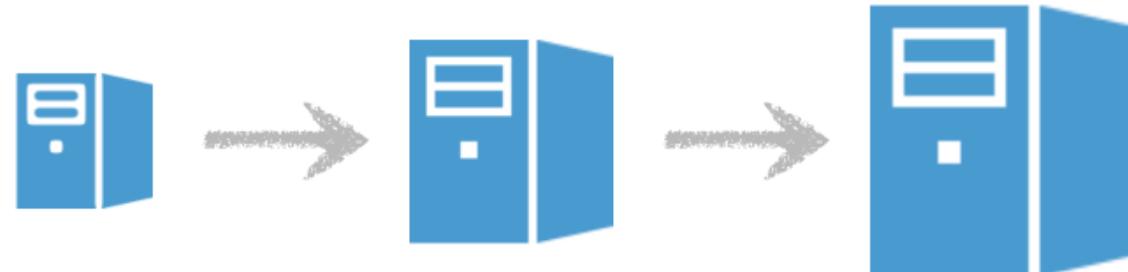
Docker: Why



Docker: Why



Scale-up and Scale-out



Scale Up



Scale Out

Image source: <https://inverodigital.com/insights/digital-transformation-the-importance-of-scalability/>

SuperAI Engineer Season#5

58

Deployment Strategies

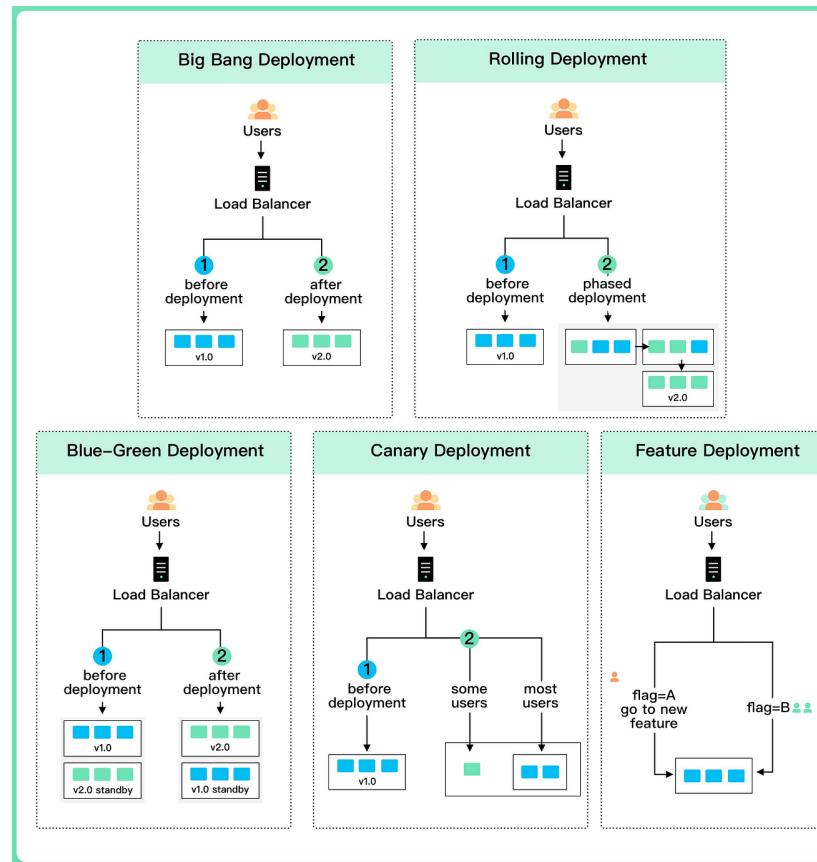


Image source: <https://harshityadav95.medium.com/notes-top-5-deployment-strategies-cc4a04ace01c>

SuperAI Engineer Season#5

59

Docker: Alternative

- <https://signoz.io/comparisons/docker-alternatives/>



podman

containerd



**RED HAT
OPENSHIFT**



RUNC



LXC



buildah



cri-o



kubernetes

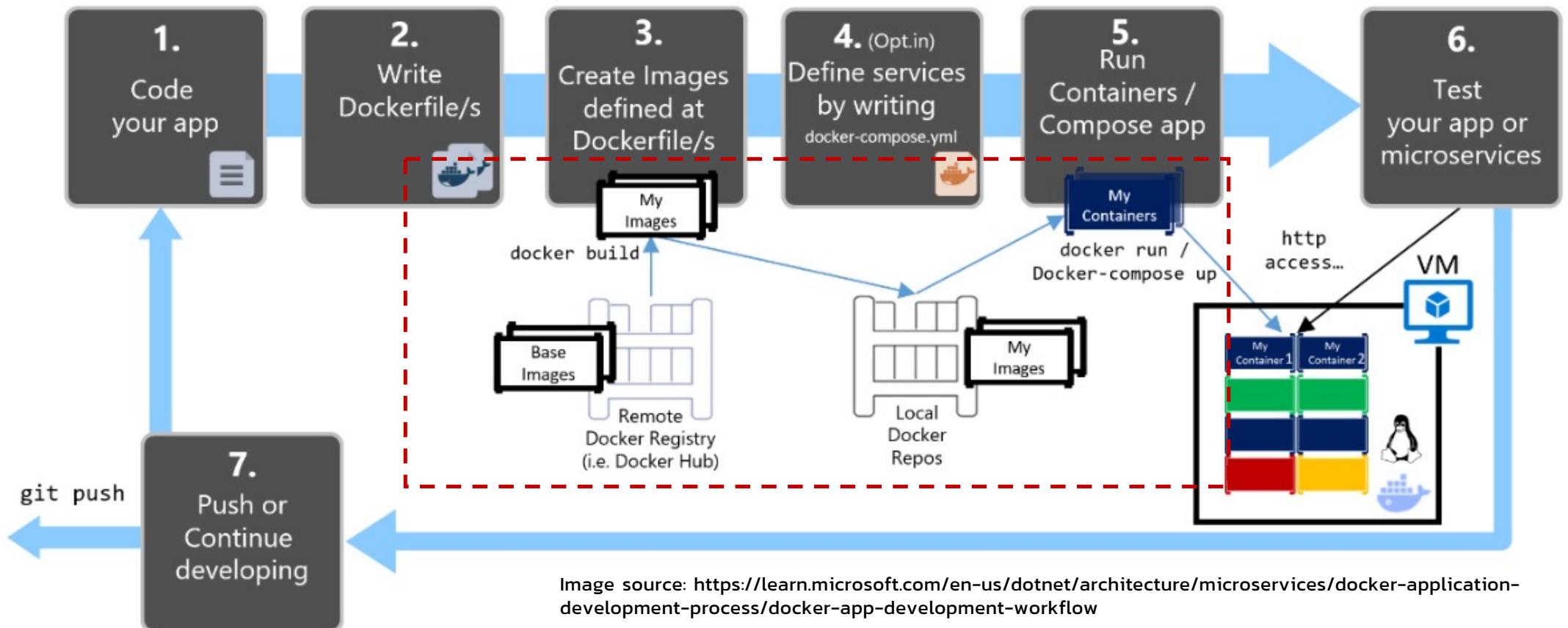


RANCHER DESKTOP

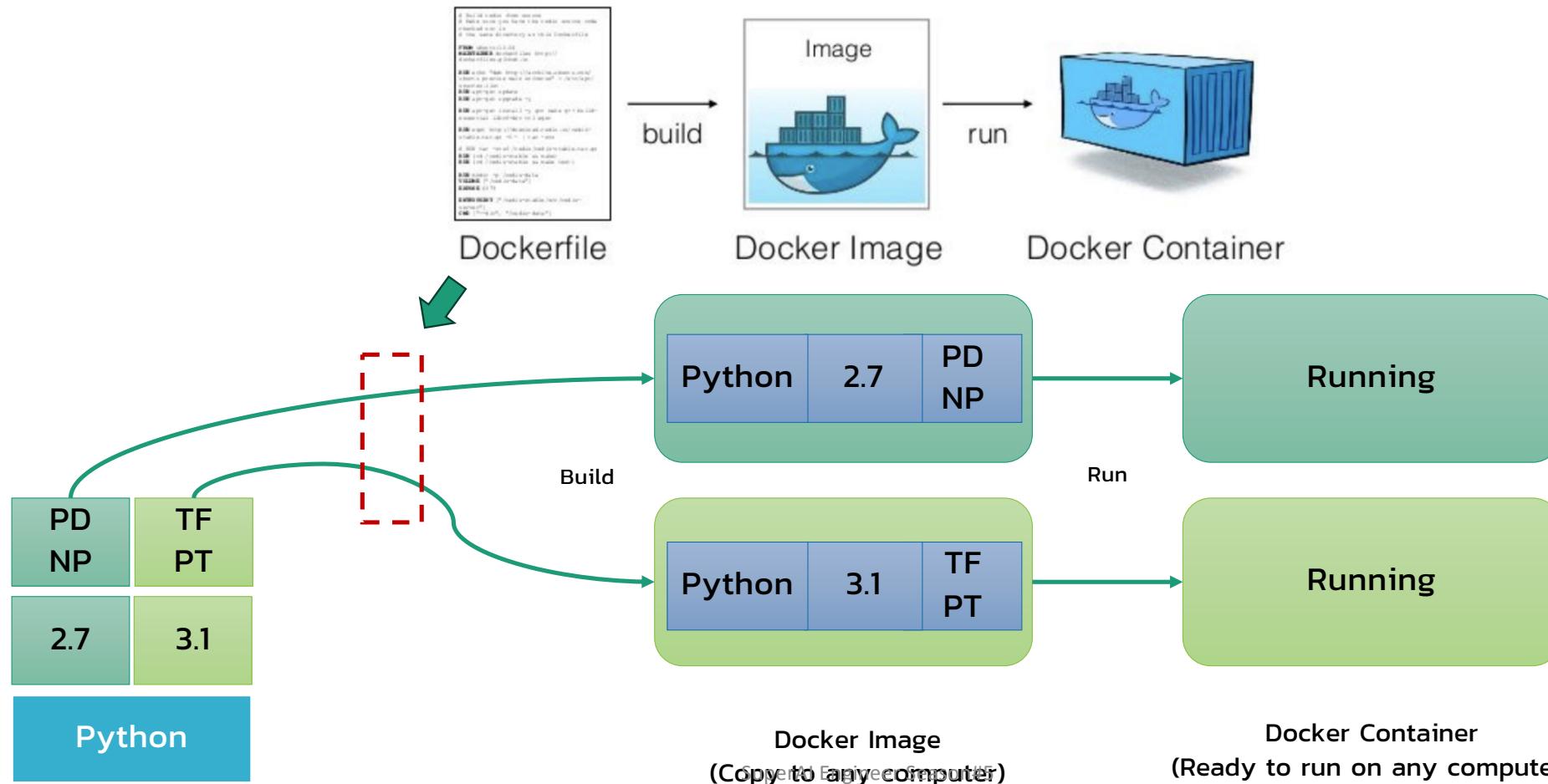


**Microsoft
Hyper-V**

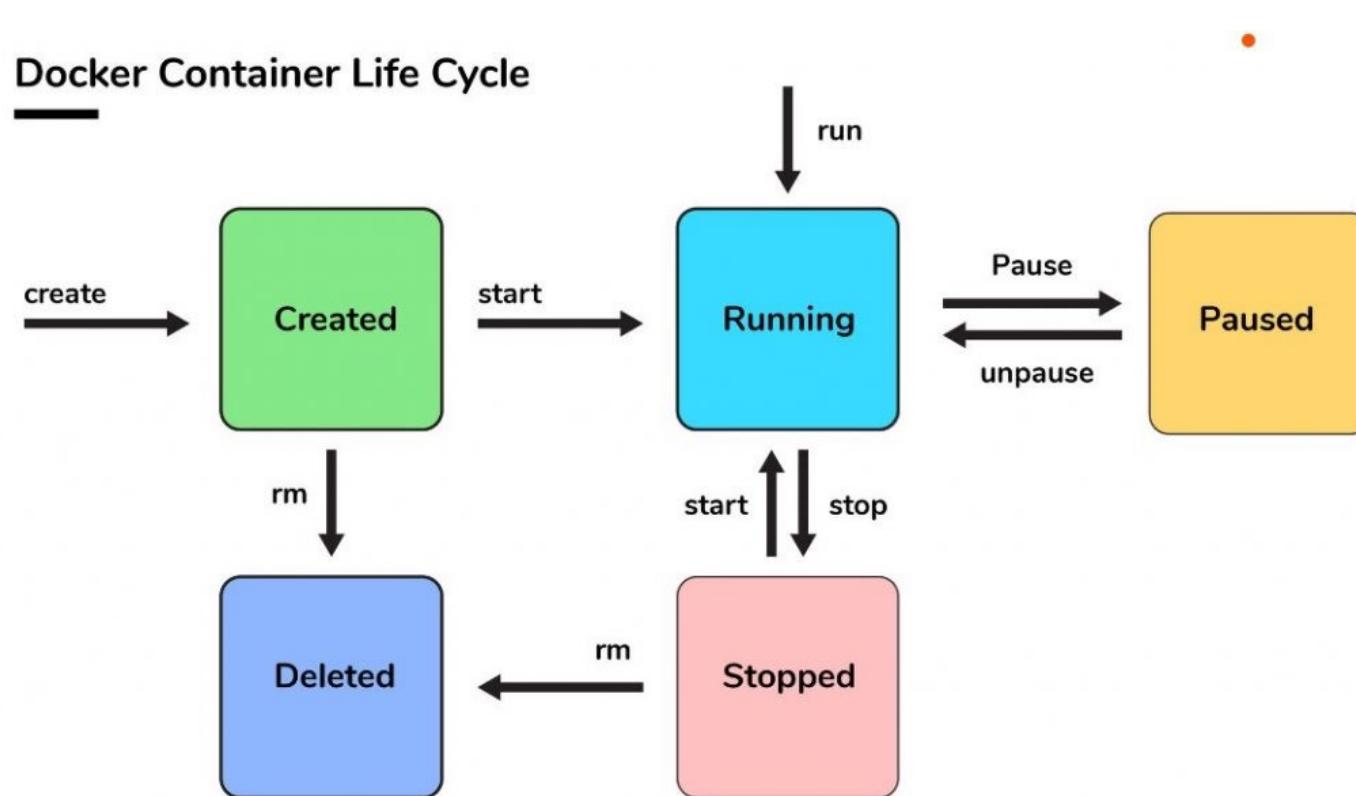
Docker: Overall application development workflow



Docker: Basic Concept



Docker: Container Life Cycle



Docker Cheat Sheet (Basic)



definitions

 image	a static snapshot of container's configuration.
 container	an application sandbox. each container is based on an image.
 layer	image is composed of read-only file system layers. container creates single writable layer.
 docker registry	remote server for storing Docker images
 Dockerfile	a configuration file with build instructions for Docker images
 docker engine	Docker platform installation running on a given host
 docker client	client application that talks to local or remote Docker daemon
 docker daemon	service process that listens to Docker client commands over local or remote network
 docker host	server that runs Docker engine
 volume	directory shared between host and container

Find more cheatsheets at <https://extremeautomation.io/cheatsheets>

docker run

```
docker run [OPTIONS] IMAGE[:TAG] [COMMAND]
Run a command in a new container.
```

metadata

- `--name=CNTR_NAME` Assign a name to the container.
- `-l, --label NAME[=VALUE]` Set metadata on the container.

process

- `-d, --detach` Run in the background.
- `-i, --interactive` Keep STDIN open.
- `-t, --tty` Allocate a pseudo-TTY.
- `--rm` Automatically remove the container when process exits.
- `-u USER` Run as username or UID.
- `--privileged` Give extended privileges.
- `-w DIR` Set working directory.
- `-e NAME=VALUE` Set environment variable.
- `--restart=POLICY` Restart policy.
 - `no` on-failure[:RETRIES]
 - `always` unless-stopped

network

- `-P, --publish-all` Publish all exposed ports to random ports.
- `-p HOST_PORT:CNTR_PORT` Expose a port or a range of ports.
- `--network=NETWORK_NAME` Connect container to a network.
- `--dns=DNS_SERVER1[,DNS_SERVER2]` Set custom dns servers.
- `--add-host=HOSTNAME:IP` Add a line to /etc/hosts.

file system

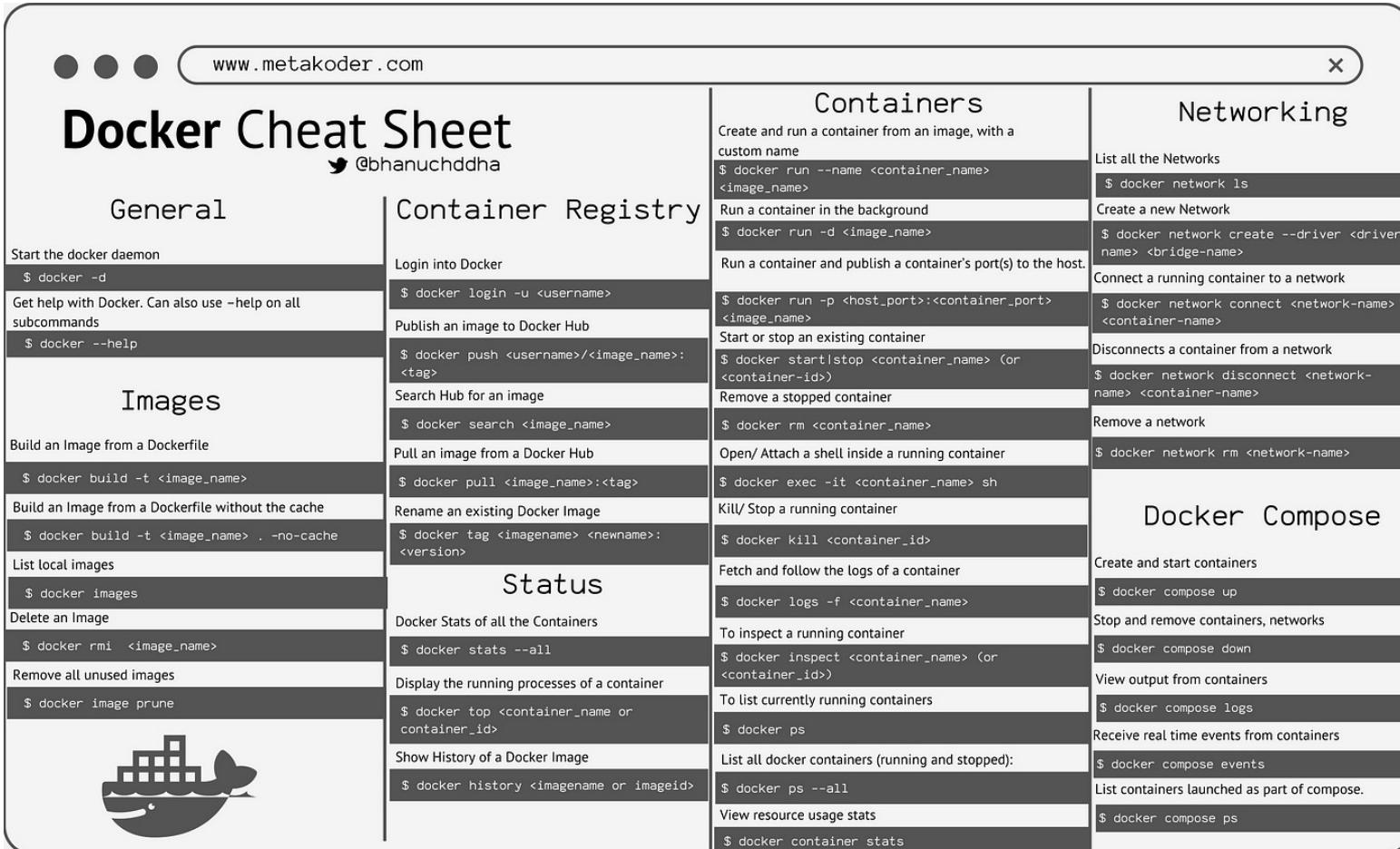
- `--read-only` Mount the container's root file system as read only.
- `-v, --volume [HOST_SRC:]CNTR_DEST` Mount a volume between host and the container file system.
- `--volumes-from=CNTR_ID` Mount all volumes from another container.

Dockerfile

```
FROM <image_id>
base image to build this image from
RUN <command> shell form
RUN ["<executable>", <param1>, exec form
     ...,
     "<paramN>"]
executes command to modify container's file system state
MAINTAINER <name>
provides information about image creator
LABEL <key>=<value>
adds searchable metadata to image
ARG <name>[=<default value>]
defines overridable build-time parameter:
docker build --build-arg <name>=<value> .
ENV <key>=<value>
defines environment variable that will be visible during image build-time and container run-time
ADD <src> <dest>
copies files from <src> (file, directory or URL) and adds them to container file system under <dest> path
COPY <src> <dest> similar to ADD, does not support URLs
VOLUME <dest>
defines mount point to be shared with host or other containers
EXPOSE <port>
informs Docker engine that container listens to port at run-time
WORKDIR <dest>
sets build-time and run-time working directory
USER <user>
defines run-time user to start container process
STOP SIGNAL <signal>
defines signal to use to notify container process to stop
ENTRYPOINT shell form or exec form
defines run-time command prefix that will be added to all run commands executed by docker run
CMD shell form or exec form
defines run-time command to be executed by default when docker run command is executed
```

ea extreme automation

Docker Cheat Sheet (Command)



The image shows a Docker Cheat Sheet (Command) page from www.metakoder.com. The page is a grid of command snippets categorized into sections: General, Images, Container Registry, Status, Containers, Networking, and Docker Compose.

General

- Start the docker daemon
\$ docker -d
- Get help with Docker. Can also use -help on all subcommands
\$ docker --help

Images

- Build an Image from a Dockerfile
\$ docker build -t <image_name>
- Build an Image from a Dockerfile without the cache
\$ docker build -t <image_name> . -no-cache
- List local images
\$ docker images
- Delete an Image
\$ docker rmi <image_name>
- Remove all unused images
\$ docker image prune

Container Registry

- Login into Docker
\$ docker login -u <username>
- Publish an image to Docker Hub
\$ docker push <username>/<image_name>: <tag>
- Search Hub for an image
\$ docker search <image_name>
- Pull an image from a Docker Hub
\$ docker pull <image_name>:<tag>
- Rename an existing Docker Image
\$ docker tag <imagename> <newname>:<version>

Status

- Docker Stats of all the Containers
\$ docker stats --all
- Display the running processes of a container
\$ docker top <container_name or container_id>
- Show History of a Docker Image
\$ docker history <imagename or imageid>

Containers

- Create and run a container from an image, with a custom name
\$ docker run --name <container_name> <image_name>
- Run a container in the background
\$ docker run -d <image_name>
- Run a container and publish a container's port(s) to the host.
\$ docker run -p <host_port>:<container_port> <image_name>
- Start or stop an existing container
\$ docker start|stop <container_name> (or <container_id>)
- Remove a stopped container
\$ docker rm <container_name>
- Open/ Attach a shell inside a running container
\$ docker exec -it <container_name> sh
- Kill/ Stop a running container
\$ docker kill <container_id>
- Fetch and follow the logs of a container
\$ docker logs -f <container_name>
- To inspect a running container
\$ docker inspect <container_name> (or <container_id>)
- To list currently running containers
\$ docker ps
- List all docker containers (running and stopped):
\$ docker ps --all
- View resource usage stats
\$ docker container stats

Networking

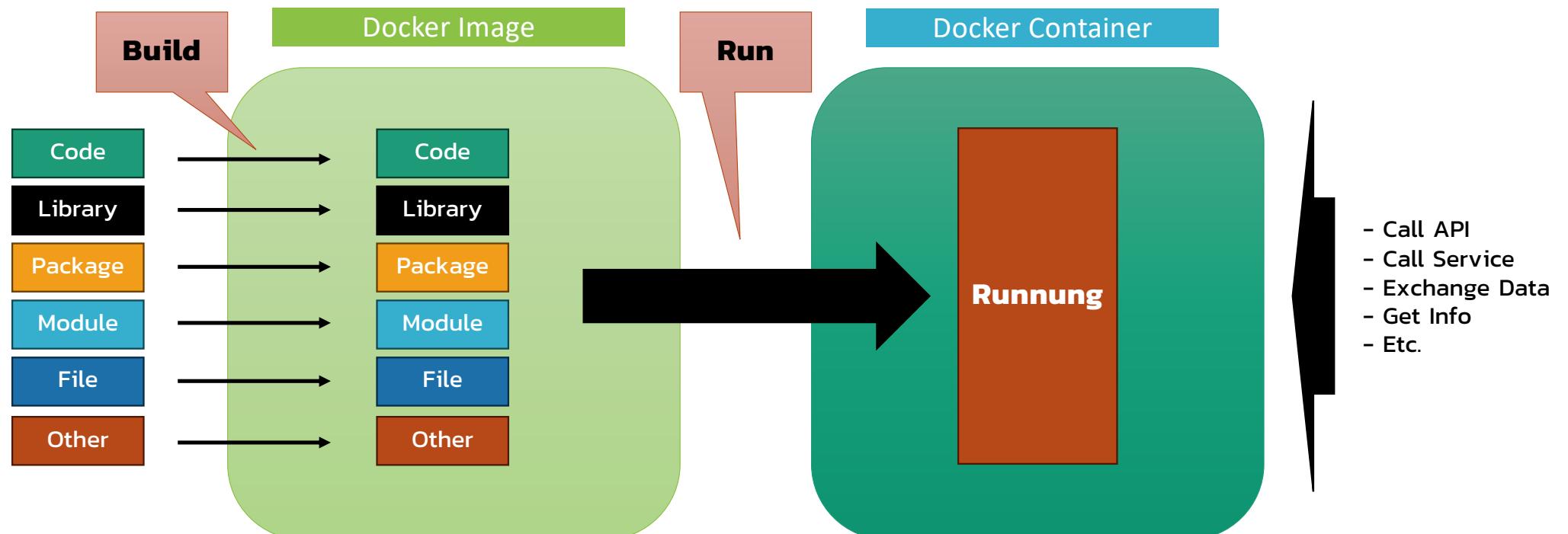
- List all the Networks
\$ docker network ls
- Create a new Network
\$ docker network create --driver <driver-name> <bridge-name>
- Connect a running container to a network
\$ docker network connect <network-name> <container-name>
- Disconnects a container from a network
\$ docker network disconnect <network-name> <container-name>
- Remove a network
\$ docker network rm <network-name>

Docker Compose

- Create and start containers
\$ docker compose up
- Stop and remove containers, networks
\$ docker compose down
- View output from containers
\$ docker compose logs
- Receive real time events from containers
\$ docker compose events
- List containers launched as part of compose.
\$ docker compose ps

Image source: <https://medium.com/@bhanuchaddha/unlocking-docker-the-ultimate-cheat-sheet-and-guide-79c3dfdee233>

Wrap-up Docker Flow



Basic Docker Command

Windows user	-> WSL Terminal
Mac user	-> Terminal
Linux user	-> Terminal

- List Docker images
 - sudo docker images

```
pongier@Windows11:~$ sudo docker images
[sudo] password for pongier:
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	74cc54e27dc4	3 months ago	10.1kB

- List Docker container
 - sudo docker ps -a

```
pongier@Windows11:~$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
89c38ee218df        hello-world        "/hello"           3 days ago        Exited (0) 3 days ago   inspiring_shamir
```

- Useful command
 - sudo docker inspect CONTAINER
 - sudo docker logs CONTAINER
 - sudo docker stat

Start with Readymade Docker image

- Meet with Docker Hub
 - <https://hub.docker.com>
- Try busybox from Docker Hub
 - sudo docker pull busybox
 - sudo docker run busybox
 - sudo docker run busybox echo "hello from busybox"
 - sudo docker run -it busybox sh
- Check Docker image/Check Docker container

Start with Readymade Docker image

```
pongier@Windows11:~$ sudo docker pull busybox
[sudo] password for pongier:
Using default tag: latest
latest: Pulling from library/busybox
97e70d161e81: Pull complete
Digest: sha256:37f7b378a29ceb4c551b1b5582e27747b855bbfaa73fa11914fe0df028dc581f
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
pongier@Windows11:~$ sudo docker run busybox
pongier@Windows11:~$ sudo docker run busybox echo "hello from busybox"
hello from busybox
pongier@Windows11:~$ sudo docker run -it busybox sh
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ # exit
pongier@Windows11:~$ |
```

Basic Docker Command

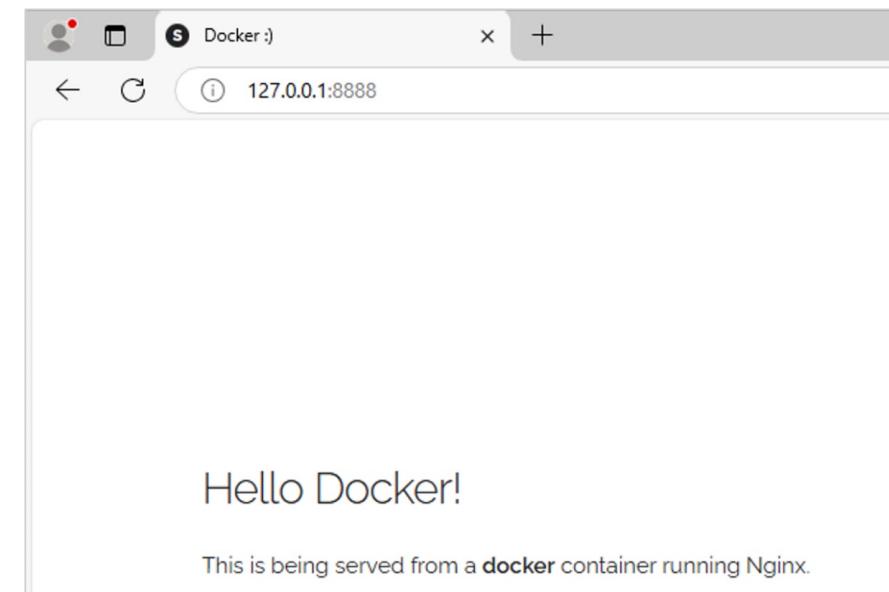
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f3f40c4a80b7	busybox	"sh"	3 minutes ago	Exited (0) 3 minutes ago		focused_germa
in						
86b88b0bf112	busybox	"echo 'hello from bu..."	3 minutes ago	Exited (0) 3 minutes ago		tender_gould
8ee177988d90	busybox	"sh"	4 minutes ago	Exited (0) 4 minutes ago		optimistic_mc
clintock						
89c38ee218df	hello-world	"/hello"	3 days ago	Exited (0) 3 days ago		inspiring_sha
mir						

- Remove Docker container
 - `sudo docker rm CONTAINER`
 - `sudo docker rm f3f40c4a80b7`
- Remove Docker image
 - `sudo docker rmi IMAGE`
 - `sudo docker rmi hello-world`

Example of running web with docker

- Readymade Docker named “prakhar1989/static-site”
 - sudo docker pull prakhar1989/static-site
 - sudo docker run -p **8888:80** prakhar1989/static-site
 - Try to open your browser with <http://127.0.0.1:8888/>

```
pongier@Windows11:~$ sudo docker run -p 8888:80 prakhar1989/static-site
Nginx is running...
```



Improvement Docker Running

- Exit from current Docker Container running

- Ctrl + c

- Find and Remove above

- sudo docker ps -a
 - sudo docker rm CONTAINER

```
pongier@Windows11:~$ sudo docker ps -a
CONTAINER ID   IMAGE           COMMAND      CREATED        STATUS          PORTS
 NAMES
8f9f925f364a   prakhar1989/static-site   "./wrapper.sh"
38 seconds ago  Exited (0) 13 seconds ago
sleepy_visvesvaraya
f3f40c4a80b7   busybox          "sh"
10 hours ago   Exited (0) 10 hours ago
focused_germain
86b88b0bf112   busybox          "echo 'hello from bu..."
10 hours ago   Exited (0) 10 hours ago
tender_gould
8ee177988d90   busybox          "sh"
10 hours ago   Exited (0) 10 hours ago
optimistic_mcclintock
89c38ee218df   hello-world       "/hello"
3 days ago     Exited (0) 3 days ago
inspiring_chain
pongier@Windows11:~$ sudo docker rm 8f9f925f364a
8f9f925f364a
```

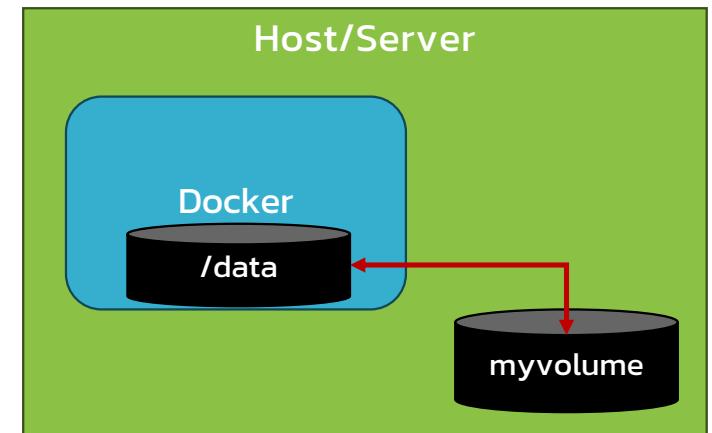
- Improve by running command (-d is Detach)

- sudo docker run -d -p 8888:80 prakhar1989/static-site

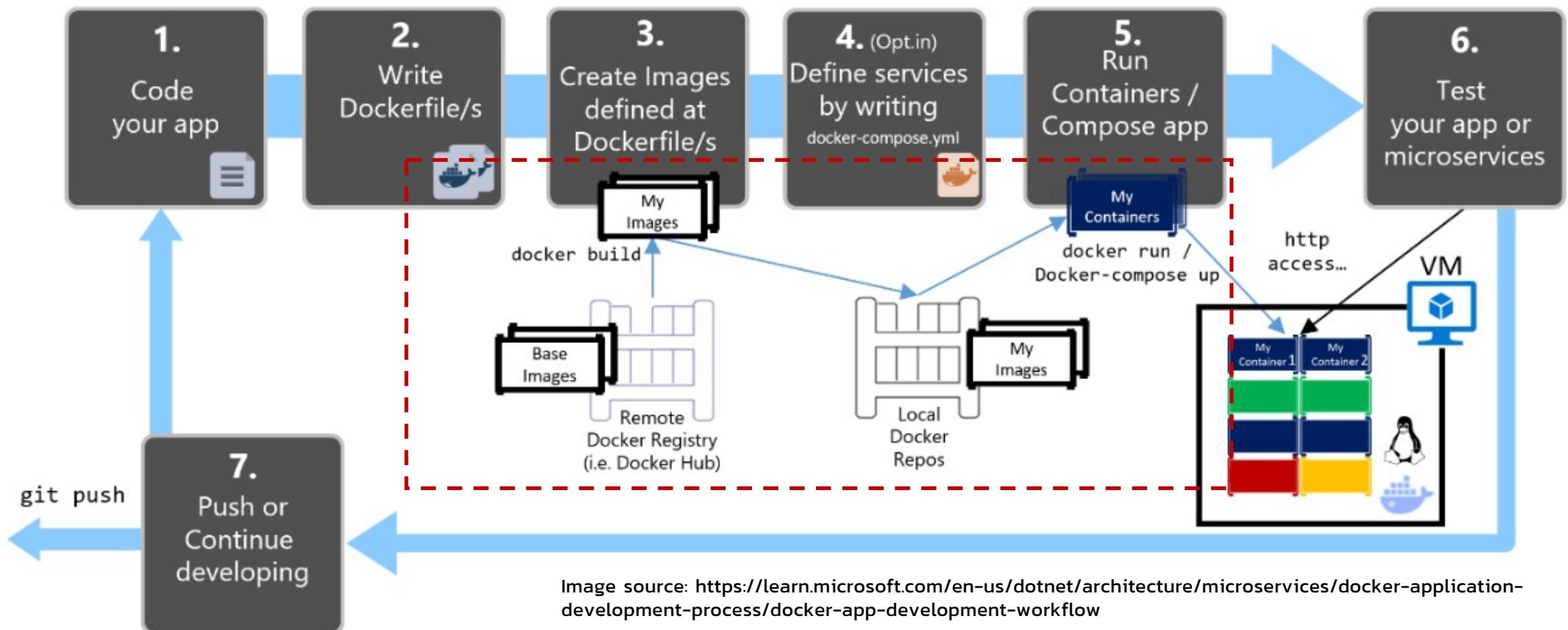
```
pongier@Windows11:~$ sudo docker run -d -p 8888:80 prakhar1989/static-site
f20f4ca9b4f8754d917b2b9ba6103c07d91323bdac16ab17e4c6bd3bd9a35ea1
```

Also Storage/Volume Mapping

- Because Docker Container is not Persistant
 - When container remove all change data is loooooose!!!
- Understanding command (-v is Volume)
 - `sudo docker run -d -v myvolume:/data example/docker`
 - myvolume is Host/Server created volume
 - /data is Container volume



Wrap-up: Overall application development workflow



Build Your Own Docker Image

- Code your app or Get some code

Windows User: (Working on WSL Terminal)

```
cd /mnt/c/API01/
```

Mac User : (Working on MacOSX Terminal)

```
cd /home/YOUR_NAME/API01/
```

Linux User : (Working on Linux OS Terminal)

```
cd /home/YOUR_NAME/API01/
```

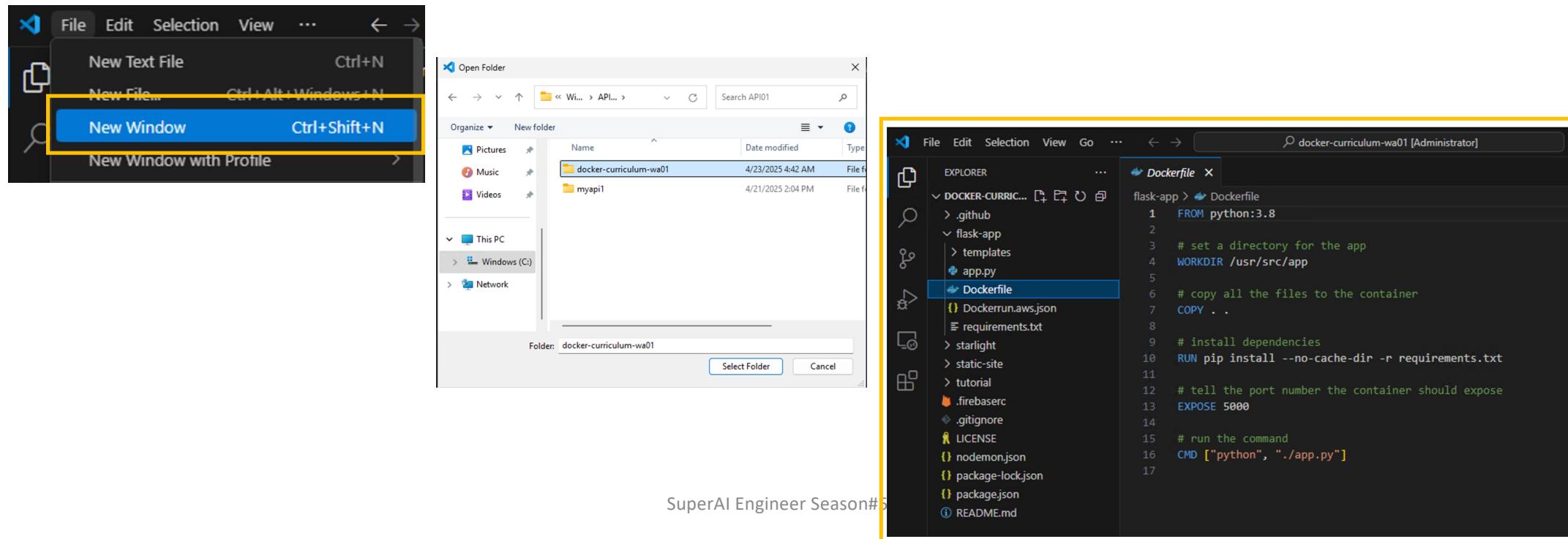
Build Your Own Docker Image

- 1. Code your app or Get some code
 - FOLLOWING YOUR OS USER COMMAND
 - `git clone https://github.com/pongier/docker-curriculum-wa01.git`
 - `cd docker-curriculum-wa01/flask-app/`

```
pongier@Windows11:~$ cd /mnt/c/API01/
pongier@Windows11:/mnt/c/API01$ git clone https://github.com/pongier/docker-curriculum-wa01.git
Cloning into 'docker-curriculum-wa01'...
remote: Enumerating objects: 1649, done.
remote: Counting objects: 100% (57/57), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 1649 (delta 43), reused 23 (delta 23), pack-reused 1592 (from 3)
Receiving objects: 100% (1649/1649), 9.12 MiB | 5.58 MiB/s, done.
Resolving deltas: 100% (932/932), done.
Updating files: 100% (172/172), done.
pongier@Windows11:/mnt/c/API01$ cd docker-curriculum-wa01/flask-app/
pongier@Windows11:/mnt/c/API01/docker-curriculum-wa01/flask-app$ |
```

Build Your Own Docker Image

- 2. Writing your Dockerfile or Customize some Dockerfile
 - Using VSCode to open Dockerfile in docker-curriculum/flask-app



Understanding Dockerfile

```
FROM python:3.8    #Base Docker Image

# set a directory for the app
WORKDIR /usr/src/app

# copy all the files to the container
COPY . .

# install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# tell the port number the container should expose
EXPOSE 5000

# run the command
CMD ["python", "./app.py"]
```

Build Your Own Docker Image

- 3. Build your Docker Image (Defined from Dockerfile)
 - sudo docker build -t yourusername/yourapp .

```
pongier@Windows11:/mnt/c/API01/docker-curriculum-wa01/flask-app$ sudo docker build -t yourusername/yourapp .
[+] Building 8.7s (9/9) FINISHED
      docker:default
      => [internal] load build definition from Dockerfile
      => => transferring dockerfile: 340B
      0.1s
      0.0s
```

- sudo docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
yourusername/yourapp	latest	8e548b66559b	3 minutes ago	1.01GB
<none>	<none>	485e449b3813	16 minutes ago	1.01GB
<none>	<none>	db4b44bda01e	34 minutes ago	1.01GB
hello-world	latest	74cc54e27dc4	3 months ago	10.1kB
busybox	latest	ff7a7936e930	6 months ago	4.28MB
prakhar1989/static-site	latest	f01030e1dcf3	9 years ago	134MB

Run Your Own Docker Container

- 5. Docker Runnnnnnnn
 - sudo docker run -p 9999:5000 yourusername/yourapp

```
pongier@Windows11:/mnt/c/API01/docker-curriculum-wa01/flask-app$ sudo docker run -p 9999:5000 yourusername/yourapp
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * 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://172.17.0.3:5000
Press CTRL+C to quit
```

- 6. Testing by open Web browser <http://127.0.0.1:9999>



SuperAI Engineer Season#5

Checking Docker Container

- Checking something
 - `sudo docker ps -a`
- Improve something
 - `sudo docker run -d -p 9999:5000 yourusername/yourapp`

Your API with Your Docker

- Remember Requirement
 - Python Version
 - python=3.12.9
 - Required Package
 - "fastapi[standard]"
 - "uvicorn[standard]"
 - install numpy
 - pandas
 - scikit-learn
 - Port
 - 10000

Understanding requirements.txt

- requirements.txt (Create in same directory of project)
 - is a file that contains a list of packages or libraries needed to work on a project that can all be installed with the file.
- List all packages or libraries that installed in this environment
 - pip freeze

```
● (superapi) pongier@Windows11:/mnt/c/API01/myapi1$ pip freeze
annotated-types @ file:///home/conda/feedstock_root/build_artifacts/annotated-types_1733247046149/work
anyio @ file:///home/conda/feedstock_root/build_artifacts/bld/rattler-build_anyio_1742243108/work
certifi @ file:///home/conda/feedstock_root/build_artifacts/certifi_1739515848642/work/certifi
```

requirements.txt of your API

```
numpy==2.2.5
pandas
scikit-learn
fastapi[standard]
uvicorn[standard]
```

Create Dockerfile for your API

```
FROM python:3.12

# set a directory for the app
WORKDIR /usr/src/app

# copy all the files to the container
COPY . .

# install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# tell the port number the container should expose
EXPOSE 5000

# run the command
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "9000", "--workers", "2"]
```

Exec form:

```
#  Do this
CMD ["fastapi", "run", "app/main.py", "--port", "80"]
```

Shell form:

```
#  Don't do this
CMD fastapi run app/main.py --port 80
```

Building Docker Image

- Buildinggggg...
 - sudo docker build -t yourusername/yourapi .

```
pongier@Windows11:/mnt/c/API01/myapi$ sudo docker build -t yourusername/yourapi .
[+] Building 49.8s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 398B
=> [internal] load metadata for docker.io/library/python:3.12
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/python:3.12@sha256:f282c100bbcd0db4346ba7261e15272937fe89de316f0a7159168da02b6fc
=> [internal] load build context
=> => transferring context: 638B
=> CACHED [2/4] WORKDIR /usr/src/app
=> [3/4] COPY .
=> [4/4] RUN pip install --no-cache-dir -r requirements.txt
=> exporting to image
=> => exporting layers
=> => writing image sha256:7dee01d434bc6c90b1ee8ef43df50bd135411020aac6837a2274c1bcf2e17192
=> => naming to docker.io/yourusername/yourapi
```

Running

- Runninggggg...
 - `sudo docker run -p 10000:9000 yourusername/yourapi`

```
pongier@Windows11:/mnt/c/API01/myapi1$ sudo docker run -p 10000:9000 yourusername/yourapi
INFO:     Uvicorn running on http://0.0.0.0:9000 (Press CTRL+C to quit)
INFO:     Started parent process [1]
INFO:     Started server process [10]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Started server process [9]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     172.17.0.1:54388 - "POST /predicts HTTP/1.1" 200 OK
```

Testing Your API

- Testinggggg...
- Postman
- <http://127.0.0.1:10000/predicts>

The screenshot shows the Postman application interface. At the top, it displays the URL `http://127.0.0.1:10000/predicts`. Below this, the method is set to `POST` and the URL is again shown as `http://127.0.0.1:10000/predicts`. The `Body` tab is selected, and the `JSON` tab is chosen under the dropdown. The JSON payload is:

```
1 "Age": 3,  
2 "Sex": "male",  
3 "Embarked": "C"
```

Below the request, the response status is `200 OK`, time `51 ms`, and size `181 B`. The response body is also in JSON format:

```
1 "prediction": "[np.int64(0), np.int64(0), np.int64(0)]"
```

Your work, Image API and Docker

- Hint...
 - <https://docs.leonardo.ai/docs/getting-started>
 - <https://github.com/enriquecatala/fastapi-ai-template>
 - <https://pub.aimind.so/building-a-image-generator-with-stable-diffusion-fastapi-and-modern-web-tech-3b5ff2b37c71>

Something in Advanced

- Security
 - HTTPS
 - API Authentication/Authorization/Accounting
 - CORS
- Running on startup
- Restarts
- Multi Workers (the number of processes running)
- More advanced...In the real world.

See More...

- Docker
 - <https://docs.docker.com/manuals/>
- Docker Tutorial
 - <https://docker-curriculum.com>
- Play With Docker
 - <https://labs.play-with-docker.com/>
- Docker Compose (Recommend)
 - <https://docs.docker.com/compose/>



Ngrok

Explore The World

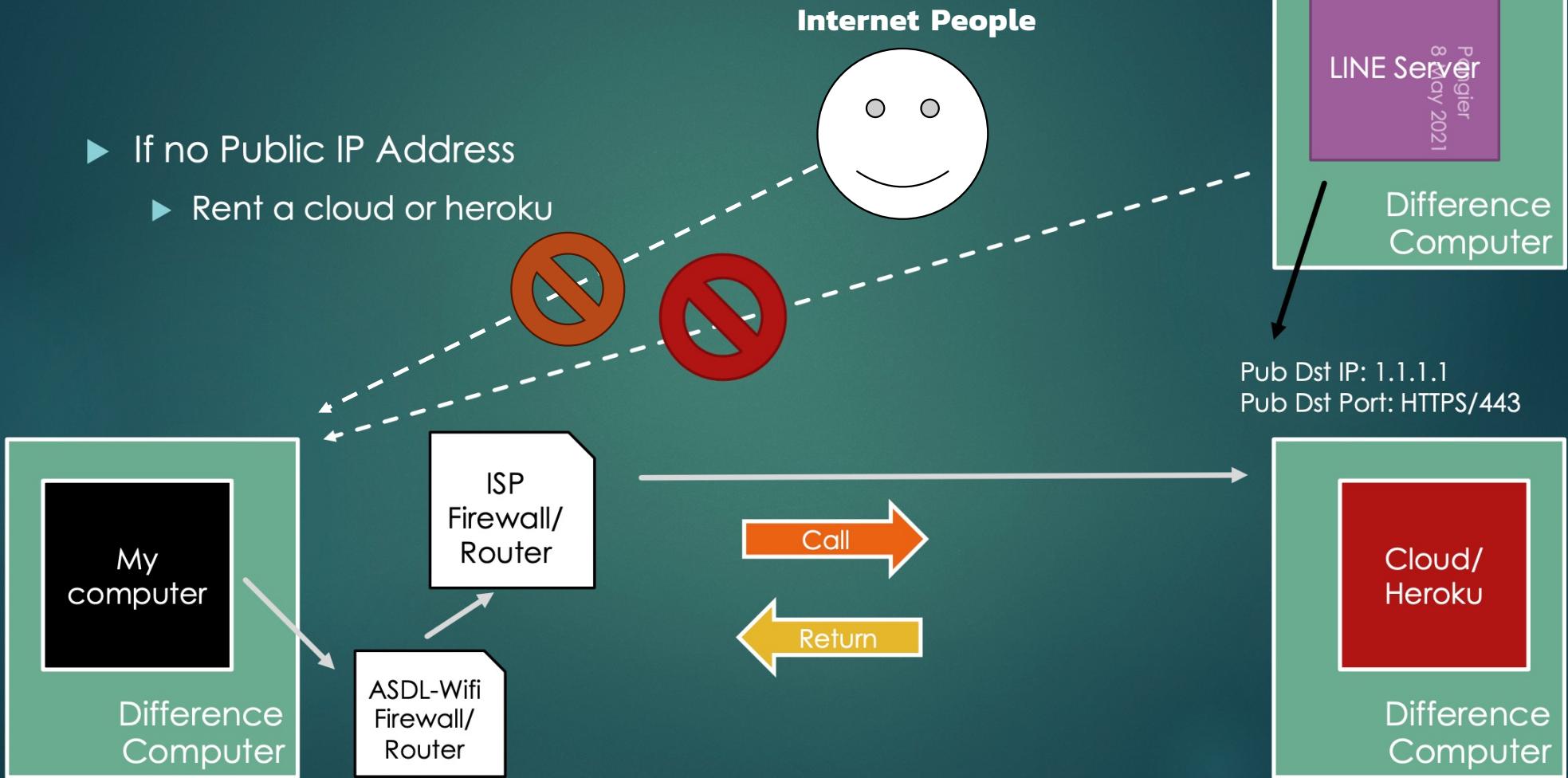
- Only have Home Internet (e.g. ADSL)
- But need Internet People could access your Application/API
- Beware!!!
 - Internet People in the world can see you.

Computer Network Concept

13

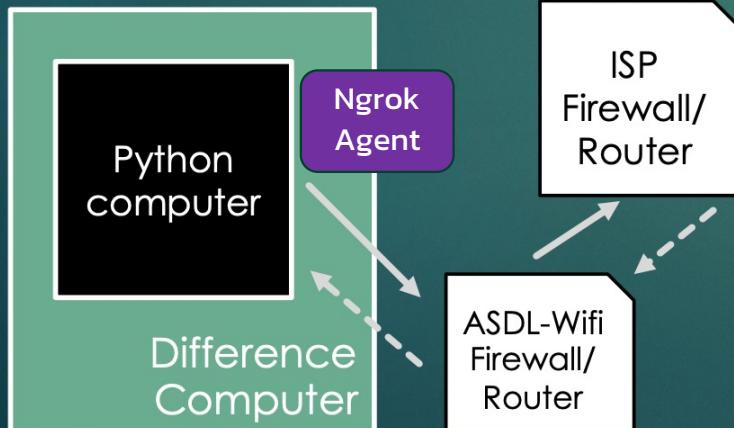
Po
8 May 2021

Difference Computer



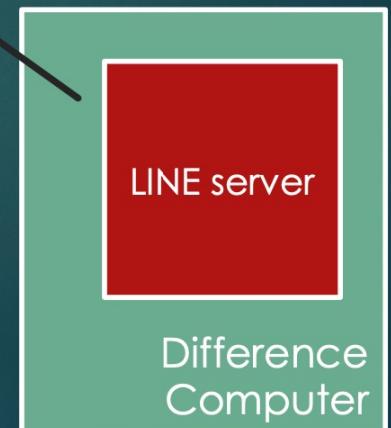
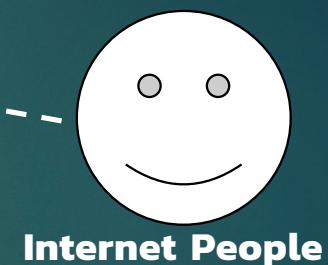
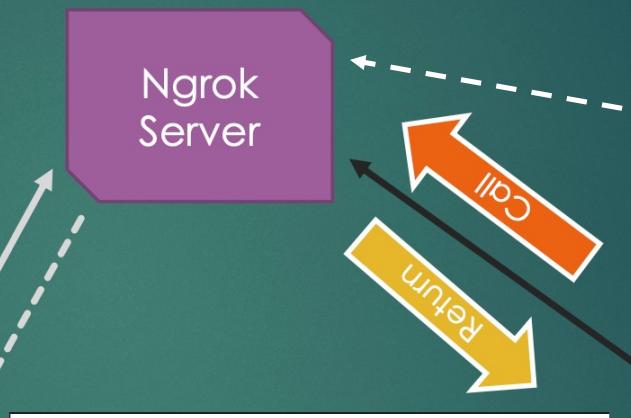
Computer Network Concept

My Computer IP: 192.168.1.10
 Python API Port: HTTP/60000



Ngrok Pub Src IP: 203.204.205.99 (Assign with name)
 Ngrok Pub Src Port: HTTP/10000 (Assign per source computer)

- ▶ If no Public IP Address
 - ▶ NGROK may by helped you.



Ngrok might help you

- Only have Home Internet (e.g. ADSL)
 - <https://ngrok.com>

The screenshot shows the ngrok pricing page with a dark header and navigation bar. The main content area features two sections: "ngrok for production" (usage-based) and "ngrok for development" (seat-based). Below these are four pricing plans:

Plan	Type	Cost	Description
FREE	Usage-based (Production)	-	Run your pre-release versions or internal apps on ngrok. Free forever.
PERSONAL	Seat-based (Development)	\$8 per developer / mo (\$10 billed monthly)	For individual developers with non-commercial projects and custom domains
PRO	Seat-based (Development)	\$20 per developer / mo (\$25 billed monthly)	For teams of developers who want to collaborate on projects
ENTERPRISE	Seat-based (Development)	\$39 per developer / mo (\$47 billed monthly)	For organizations with large engineering teams and security needs

Each plan includes a "Get started" button and a brief description of its features. A "Help" button is located in the bottom right corner.

Make sure your API or Docker is online

- Checking
 - sudo docker ps -a

```
(superapi) pongier@Windows11:/mnt/c/API01/myapi1$ sudo docker ps -a
[sudo] password for pongier:
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
S
e5f9d913c9f5      yourusername/yourapi   "uvicorn main:app --..."   2 hours ago       Up 2 hours          5000/tcp
/tcp, 0.0.0.0:10000->9000/tcp, [::]:10000->9000/tcp   youthful_hypatia
```

Getting Start with Ngrok

- <https://ngrok.com/docs/getting-started/?os=windows>
- Register and Login for an ngrok account
 - <https://dashboard.ngrok.com/login>
- Copy your **ngrok authtoken** from your ngrok dashboard

Getting Start wit Ngrok

- Run the following command in your terminal to install the authtoken and connect the Ngrok agent to your account.
 - `ngrok config add-authToken TOKEN`
- Put your app online
 - `ngrok http http://127.0.0.1:8080`

```
ngrok
(Ctrl+C to quit) □

Session Status      online
Account            inchoate (Plan: Free)
Version             3.0.0
Region              United States (us)
Latency             78ms
Web Interface      http://127.0.0.1:4040
Forwarding          https://84c5df474.ngrok-free.dev -> http://localhost:8080

Connections        ttl     opn     rt1     rt5     p50     p90
                  0       0     0.00    0.00    0.00    0.00
```

Testing Your API same as Internet Poepole

- Testing
 - Postman
 - <https://84c5df474.ngrok-free.dev/predicts>

The screenshot shows the Postman application interface. At the top, there is a header bar with the URL <https://84c5df474.ngrok-free.dev/predicts>. Below the header, a search bar contains the text "Send". Underneath the search bar, there are tabs for "Params", "Auth", "Headers (8)", "Body", "Pre-req.", "Tests", and "Settings". The "Body" tab is currently selected and has a green dot next to it. Below the tabs, there are two dropdown menus: "raw" and "JSON". The "JSON" menu is selected and has a blue background. A large text area displays the following JSON input:

```
1
2   "Age": 3,
3   "Sex": "male",
4   "Embarked": "C"
```

Below the input area, there is a "Body" section with a dropdown menu set to "Pretty". To the right of this, there are status indicators: a globe icon, "200 OK", "51 ms", and "181 B". Next to these is a "Save Response" button. At the bottom of the body section, there are four buttons: "Pretty" (selected), "Raw", "Preview", and "Visualize". To the right of these buttons is a search icon.

On the right side of the main interface, there is a vertical sidebar with a "Beautify" button at the top. Below it, there is a "Cookies" section and a "Save" button at the very top right of the main window.

At the bottom right corner of the image, there is a small number "100".

Q&A

If any problem,
Please feel free to contact me.

SEASON 5

SUPER AI ENGINEER

AI INNOVATOR • AI ENGINEER • AI RESEARCHER

