

# Docker & Deep Learning Application

Mì AI

# DL Deployment

## Python Environment Setup for Machine Learning and Deep Learning on a Remote Linux Server



**Step 1: Access the remote server**

**Step 2: Download Anaconda**

**Step 3: Run the installer**

**Step 4: Create and activate conda environment**

**Step 5: Install Tensorflow**

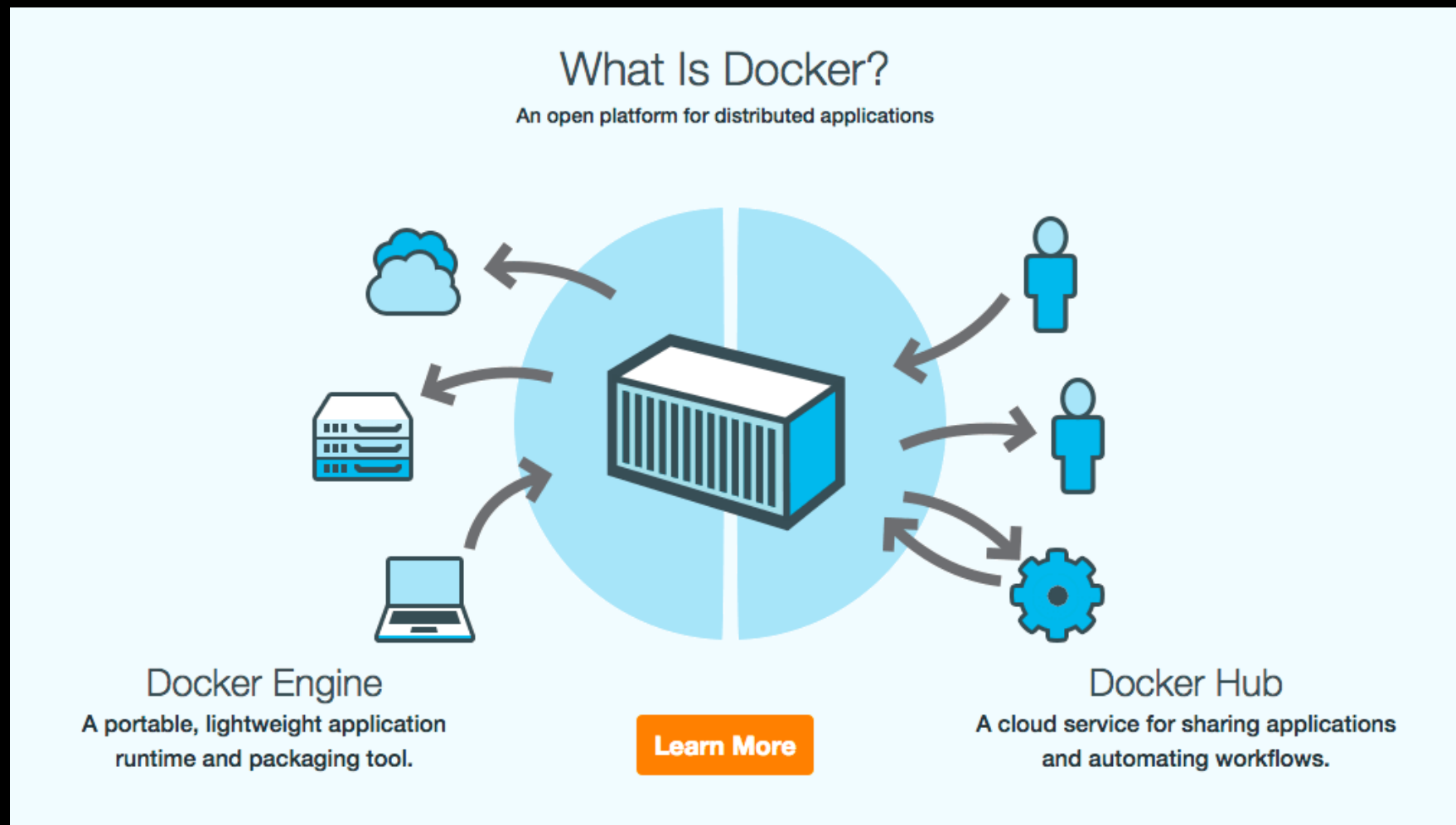
**Step 6: Testing your Tensorflow Installation**

**Step 7: Install Pandas, Jupyter Notebooks and Scikit-Learn using conda**

**Step 8: Open and access a Jupyter Notebook from the Remote Server**

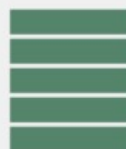
**Step 9: Open Jupyter notebook of the conda environment you created**

# Docker



# Docker

## Virtual Machine vs. Docker



Utilization



Size



Boot Up Time

VM1

VM2

VM3

APP 1

APP 2

APP 3

Bins/libs

Bins/libs

Bins/libs

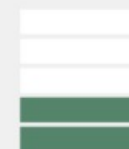
Guest OS

Guest OS

Guest OS

Hypervision

Physical Server



Utilization



Size



Boot Up Time

Container 1

Container 2

Container 3

APP 1

APP 2

APP 3

Bins/libs

Bins/libs

Bins/libs

Docker Engine

Operating System (Host OS)

Physical Server or Edge Device

# Docker

## Advantages of Docker

Isolation  
of apps



OS  
Portability



Component  
Composability



Scaling &  
Orchestration



# Docker



The slide features a dark background with a blue and white abstract shape in the center. The text is in Vietnamese. At the top right, it says 'Thực hành docker, luôn và ngay!' in yellow. On the left, it asks 'DOCKER LÀ CÁI...CHI CHI?' in orange and white. In the center, it says 'Một số lệnh docker' in white. On the right, there is a list of Docker commands in white text.

**Thực hành docker, luôn và ngay!**

**DOCKER**  
**LÀ CÁI...CHI CHI?**

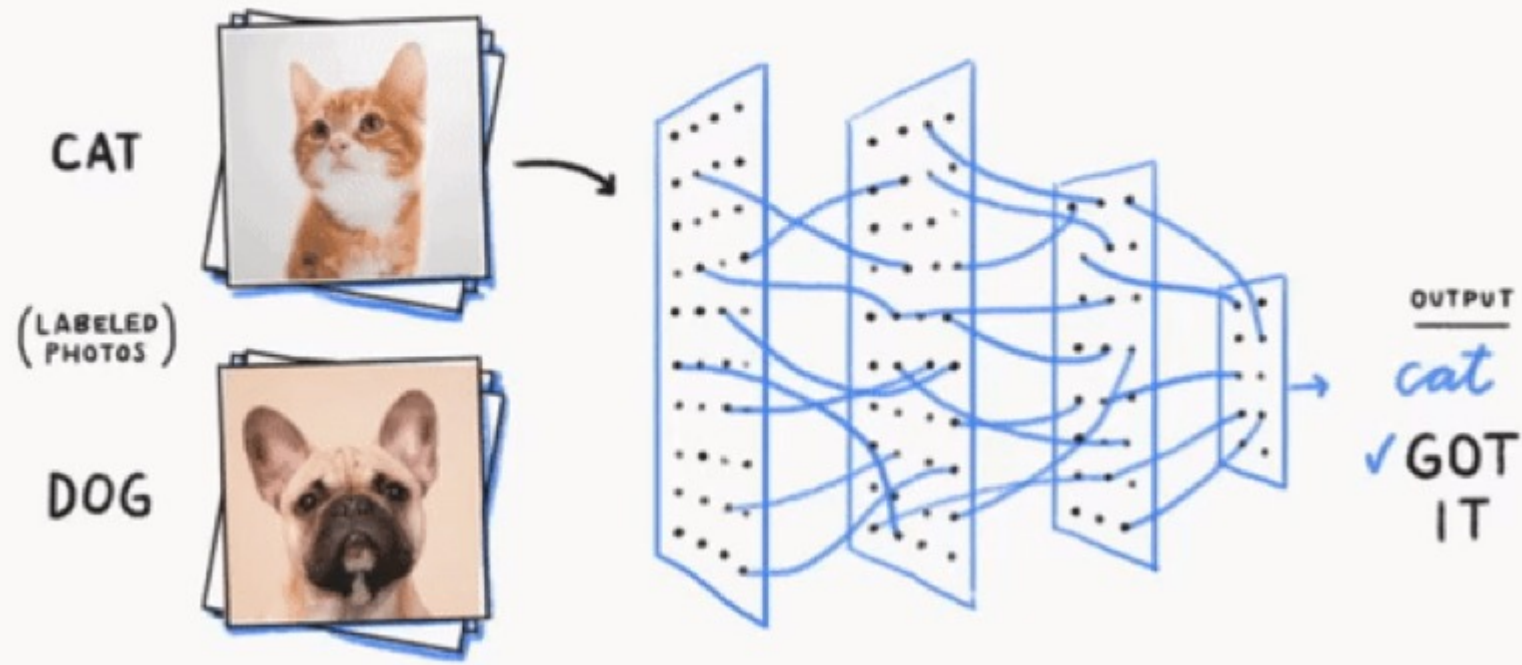
Một số lệnh docker

- docker search
- docker pull
- docker build
- docker images
- docker run
- docker container
- docker network

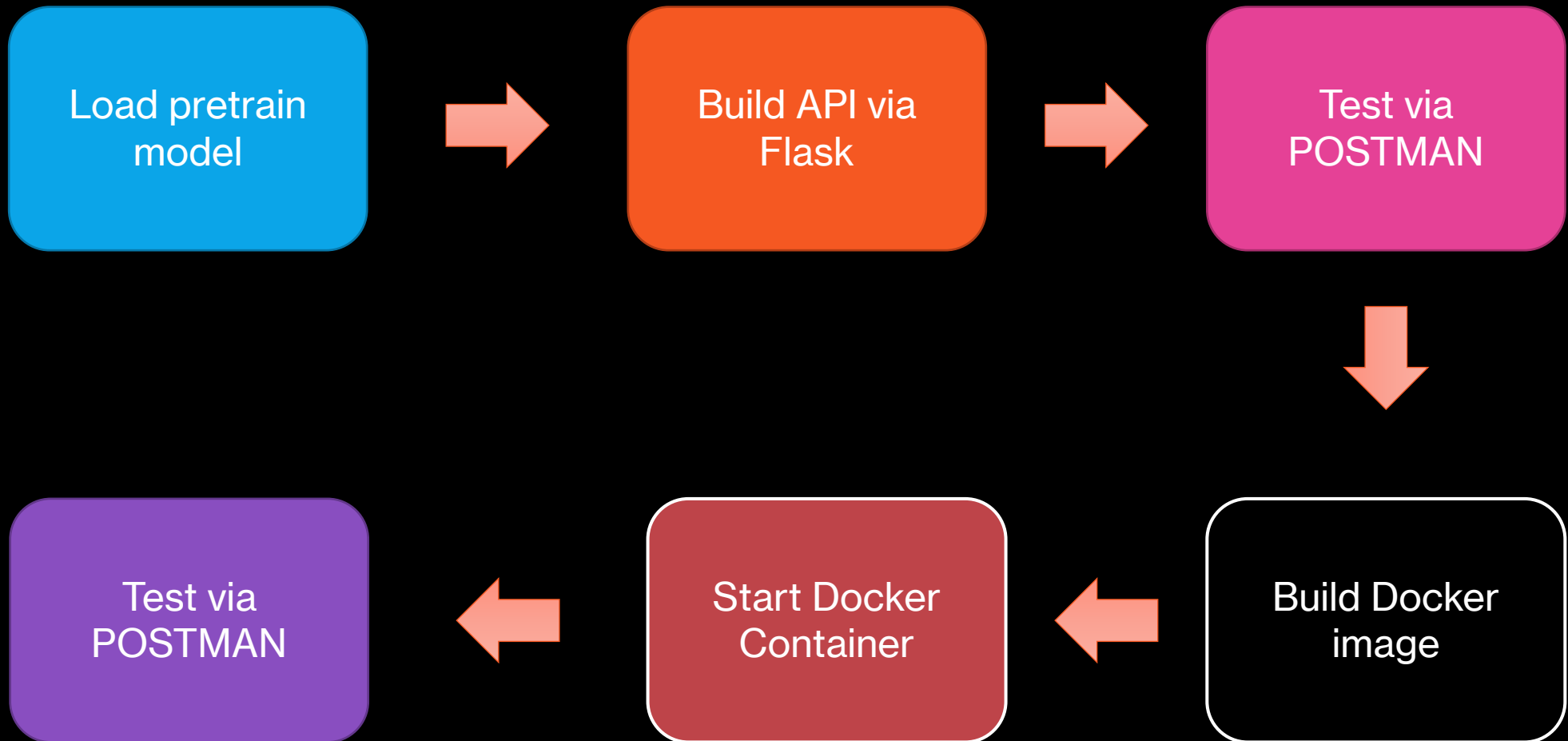


# App of today

A Neural Network is a function that can learn



# Pipeline





# Handson

```
31 def __init__(self, path):
32     self.file = None
33     self.fingerprints = set()
34     self.logdups = True
35     self.debug = debug
36     self.logger = logging.getLogger(__name__)
37     if path:
38         self.file = open(os.path.join(path, 'requests.log'),
39                           'a')
40         self.file.seek(0)
41         self.fingerprints.update(self.retrieve())
42
43 @classmethod
44 def from_settings(cls, settings):
45     debug = settings.getbool('SUPERFILTER_DEBUG')
46     return cls(job_dir(settings), debug)
47
48 def request_seen(self, request):
49     fp = self.request_fingerprint(request)
50     if fp in self.fingerprints:
51         return True
52     self.fingerprints.add(fp)
53     if self.file:
54         self.file.write(fp + os.linesep)
55
56 def request_fingerprint(self, request):
57     return request_fingerprint(request)
```