UNIVERSITAT DE BARCELONA

Master in Fundamental Principles of Data Science

Dr Rohit Kumar

UNIVERSITAT DE BARCELONA

Today's Objective

- Learn to dockerize your models
- Multiple Container orchestration.

UNIVERSITAT DE BARCELONA

Docker

UNIVERSITAT DE BARCELONA

What is Docker

Docker is an operating system **container** management tool that allows you to easily manage and deploy applications by making it easy to package them within operating system containers. Docker's portability and lightweight also make it easy to dynamically manage workloads, scaling up or tearing down applications, in near real time.One of the main benefits of using Docker, and container technology, is the portability of applications. It's possible to spin up an application on-prem or in a public cloud environment in a matter of minutes.

UNIVERSITAT DE BARCELONA

In Simpler words

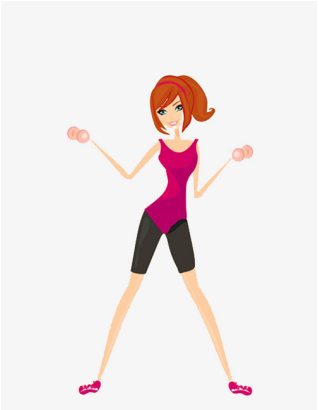
Docker is a tool that allows developers, sys-admins etc. to easily deploy their applications in a sandbox (called ***containers***) to run on the host operating system i.e. Linux. The key benefit of Docker is that it allows users to **package an application with all of its dependencies into a standardized unit** for software development.

UNIVERSITAT DE BARCELONA

What are containers

Virtual Machines- Good isolation but a higher computational overhead.

Containers on the other hand take a different approach: by leveraging the low-level mechanics of the host operating system, containers provide most of the isolation of virtual machines at a fraction of the computing power.

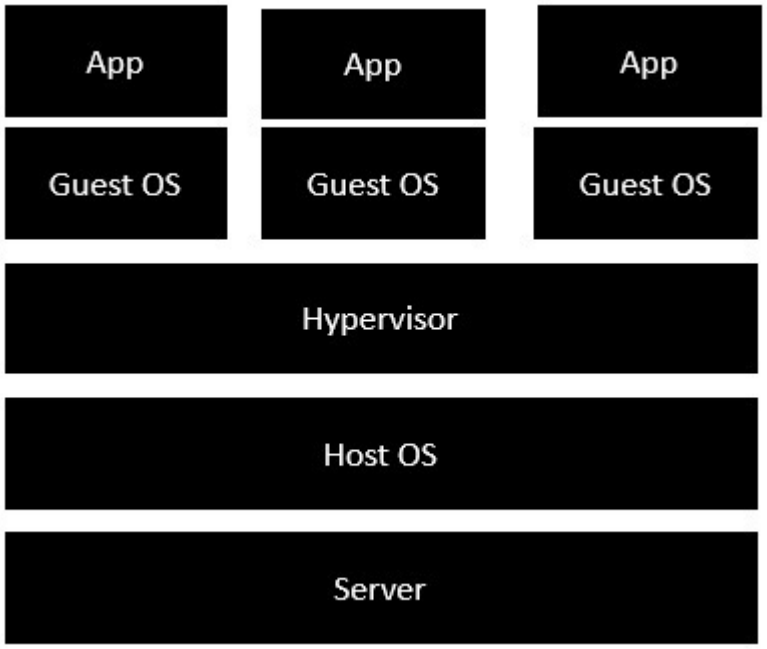


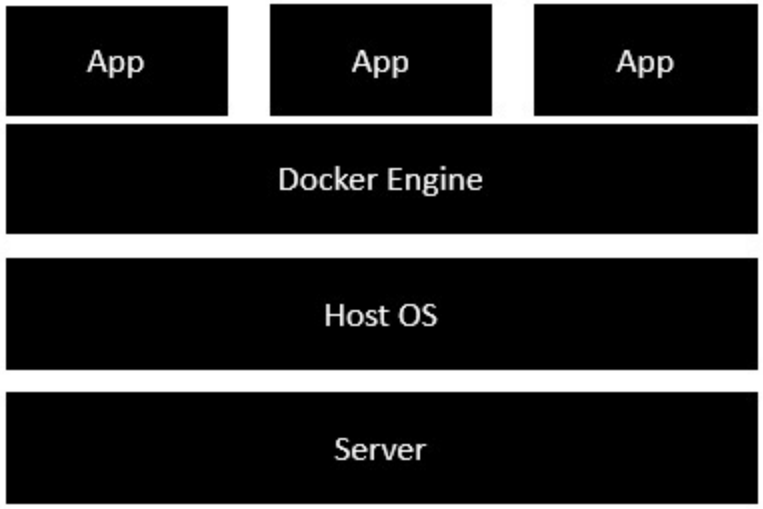
For now, you can think of a container

as a lightweight equivalent of a virtual machine.

UNIVERSITAT DE BARCELONA

VM vs Docker





UNIVERSITAT DE BARCELONA

Why Containers

- They are slim !!
- Docker enables developers to easily pack, ship, and run any application as a lightweight, portable, self-sufficient container, which can run virtually anywhere.
- It Makes CI/CD very much easy.
- Using orchestration tools like Kubernetes they make scalability much easy and fast on cloud.

UNIVERSITAT DE BARCELONA

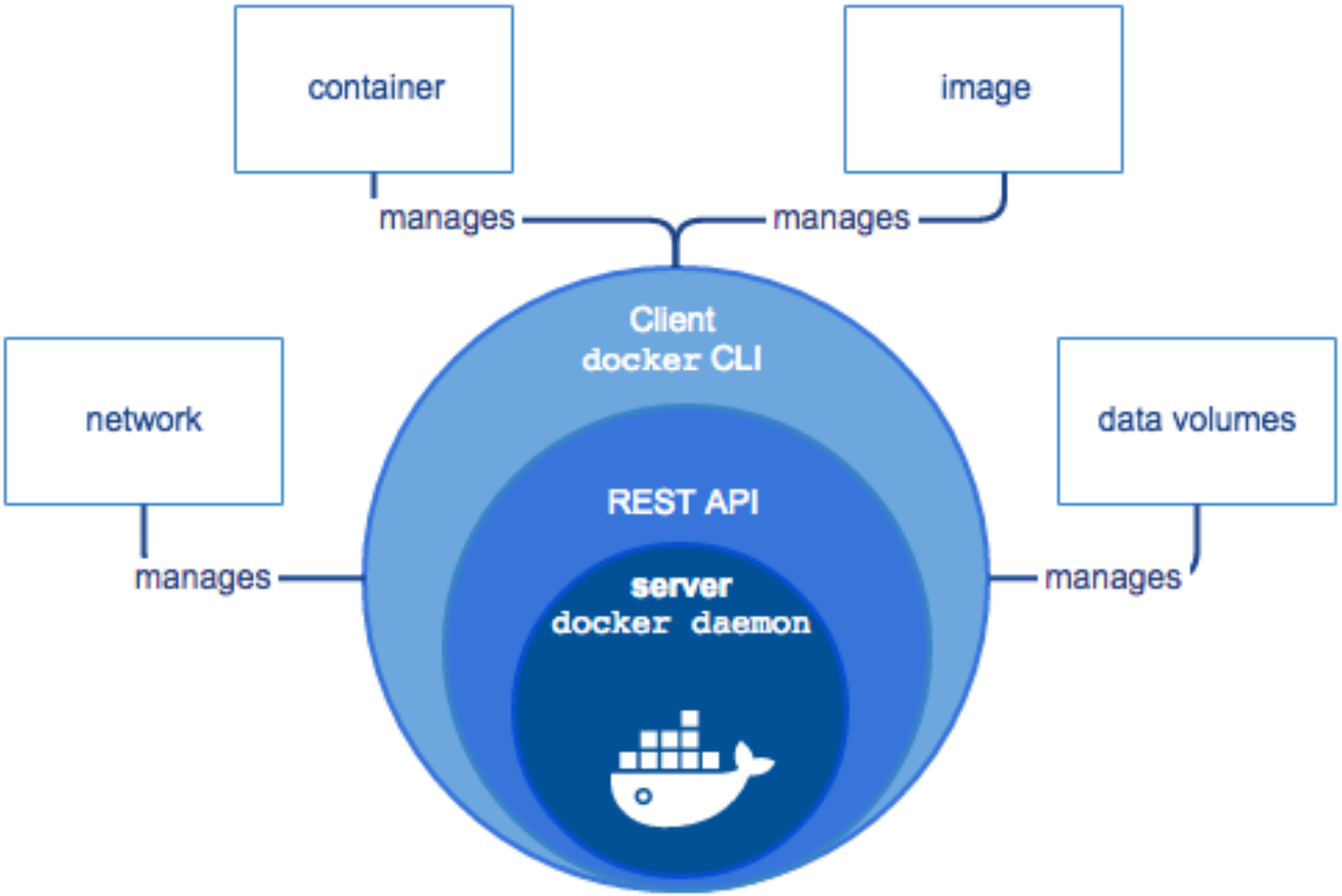
Docker Engine

Docker Engine is a client-server application with these major components:

- A **server** which is a type of long-running program called a daemon process (the dockerd command).
- A **REST API** which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
- A command line interface (**CLI**) client (the docker command).

UNIVERSITAT DE BARCELONA

Docker Engine



UNIVERSITAT DE BARCELONA

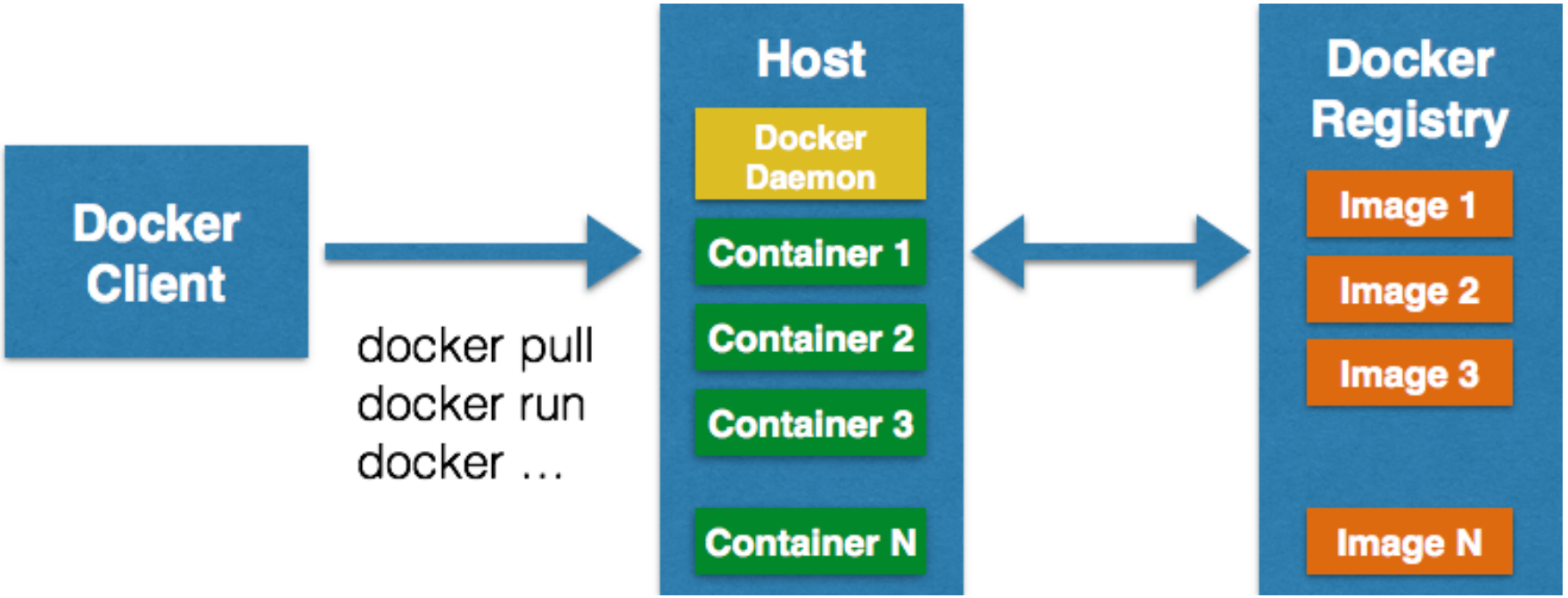
Docker architecture

Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker objects such as images, containers, network and volumes.

The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.

UNIVERSITAT DE BARCELONA

Docker architecture



UNIVERSITAT DE BARCELONA

Docker architecture

Docker Images

Docker images are the template for our containers; we use them to build containers. They can have software pre-installed which speeds up deployment. They are portable, and we can use existing images or build our own.

Docker Containers

Containers are the organizational units of Docker. When we build an image and start running it; we are running in a container. The container analogy is used because of the portability of the software we have running in our container.

In simple terms, an image is a template, and a container is a runnable instance of the image. You can have multiple containers (copies) of the same image.

UNIVERSITAT DE BARCELONA

Docker architecture

Registries

Docker stores the images we build in registries. There are public and private registries. Docker company has public registry called Docker hub, where you can also store images privately. Docker hub has millions of images, which you can start using now.

Create an account

https://hub.docker.com/

UNIVERSITAT DE BARCELONA

Docker Hands On

$ docker pull busybox

The pull command fetches the busybox image from the Docker registry and saves it to our system. You can use the docker images command to see a list of all images on your system.

$ docker images

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL
busybox             latest              c51f86c28340        4 weeks ago         1.109 MB
```

UNIVERSITAT DE BARCELONA

Docker Hands On

Run a Docker **container** based on this image. To do that we are going to use the docker run command.

$ docker run busybox

UNIVERSITAT DE BARCELONA

Docker Hands On

Behind the scenes, a lot of stuff happened. When you call run, the Docker client finds the image (busybox in this case), loads up the container and then runs a command in that container. When we run docker run busybox, we didn't provide a command, so the container booted up, ran an empty command and then exited.

Now Try

$ docker run busybox echo "hello from busybox"

UNIVERSITAT DE BARCELONA

Docker Hands On

List Running Containers history

$ docker ps –a

Now try naming your container

$ docker run –name bts busybox echo "hello from busybox”

$ docker ps –a

UNIVERSITAT DE BARCELONA

Docker Hands On

Running the run command with the -it flags attaches us to an interactive tty in the container.

$ docker run -it busybox sh

To exit do exit or cltr+D

$ docker run –help

to see a list of all flags run command supports.

UNIVERSITAT DE BARCELONA

Docker Hands On

Cleanup

We saw above that we can still see remnants of the container even after we've exited by running docker ps –a. Hence as a rule of thump clean up conatiners once done.

Use docker rm command to delete a container

$ docker rm <container ids>

$ docker rm df0da980903d 04b866b7454d

UNIVERSITAT DE BARCELONA

Docker Hands On

$ docker rm $(docker ps -a -q -f status=exited)

This command deletes all containers that have a status of exited. the -q flag, only returns the numeric IDs and -f filters output based on conditions provided.

You can also use --rm flag that can be passed to docker run which automatically deletes the container once it's exited from. For one off docker runs, --rm flag is very useful.

In later versions of Docker, the ***docker container prune*** command can be used to achieve the same effect.

UNIVERSITAT DE BARCELONA

Docker Hands On

You can also delete images that you no longer need by running **docker rmi**.

In later versions of Docker, the ***docker container prune*** command can be used to achieve the same effect.

And **docker image prune** to remove dangling images.

UNIVERSITAT DE BARCELONA

Docker Hands On

Now lets run Jupyter Notebook using Docker

$ docker run -p 8888:8888 jupyter/minimal-notebook

UNIVERSITAT DE BARCELONA

Docker Hands On

You see the saved notebooks are gone as soon as you stop the container.

Now try this:

- Create a local folder to save notebooks

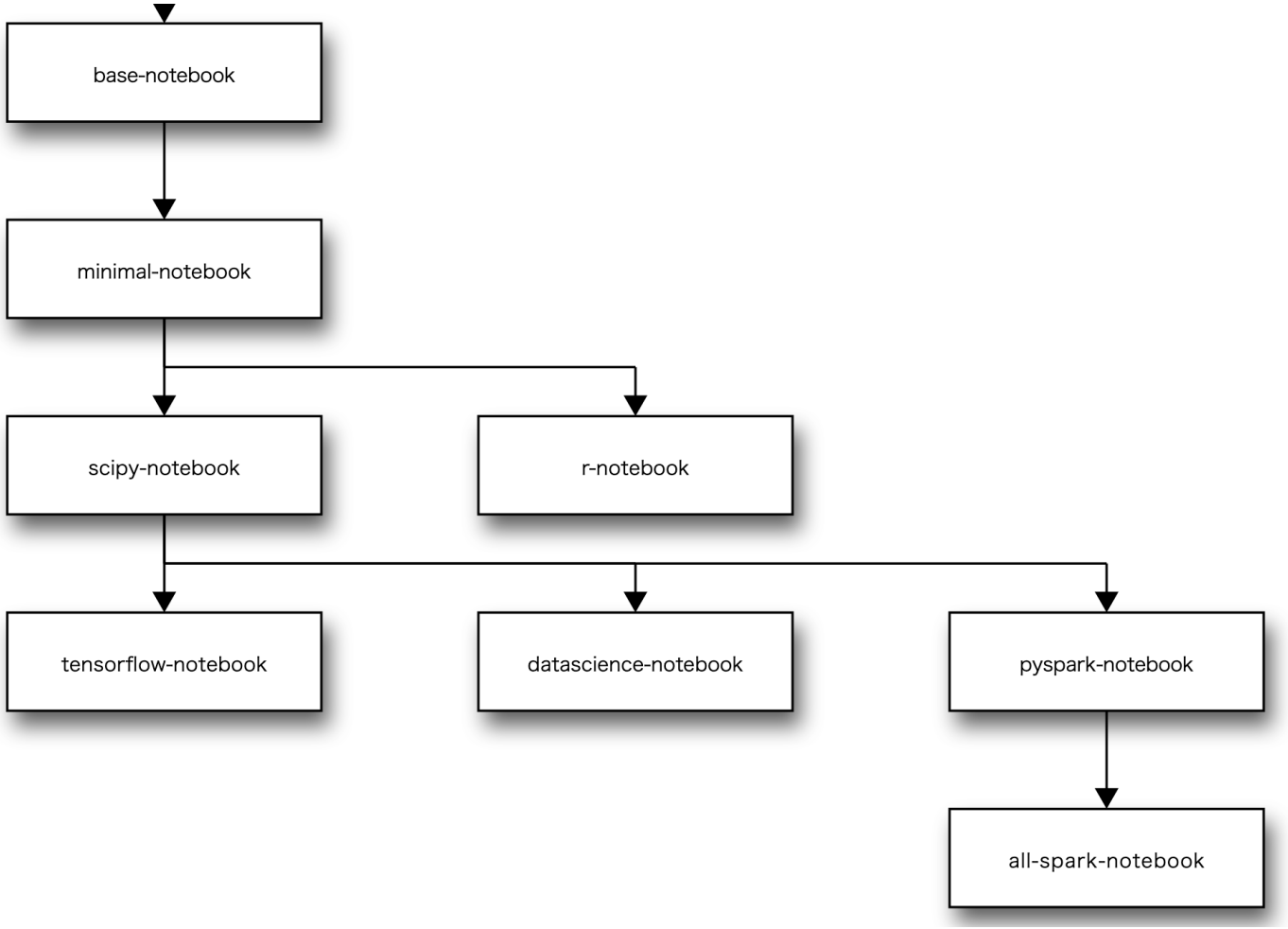
  $ mkdir ~/notebooks

- We can now share this directory between the host and container. The flag for this argument is -v <host_directory>:<container_directory> which tells the Docker engine to **mount** the given host directory to the container directory

  $ docker run -p 8888:8888 -v ~/notebooks:/home/jovyan jupyter/minimal-notebook

UNIVERSITAT DE BARCELONA

Jupyter Dockers



https://jupyter-docker-stacks.readthedocs.io/en/latest/using/selecting.html

UNIVERSITAT DE BARCELONA

Manage data in Docker

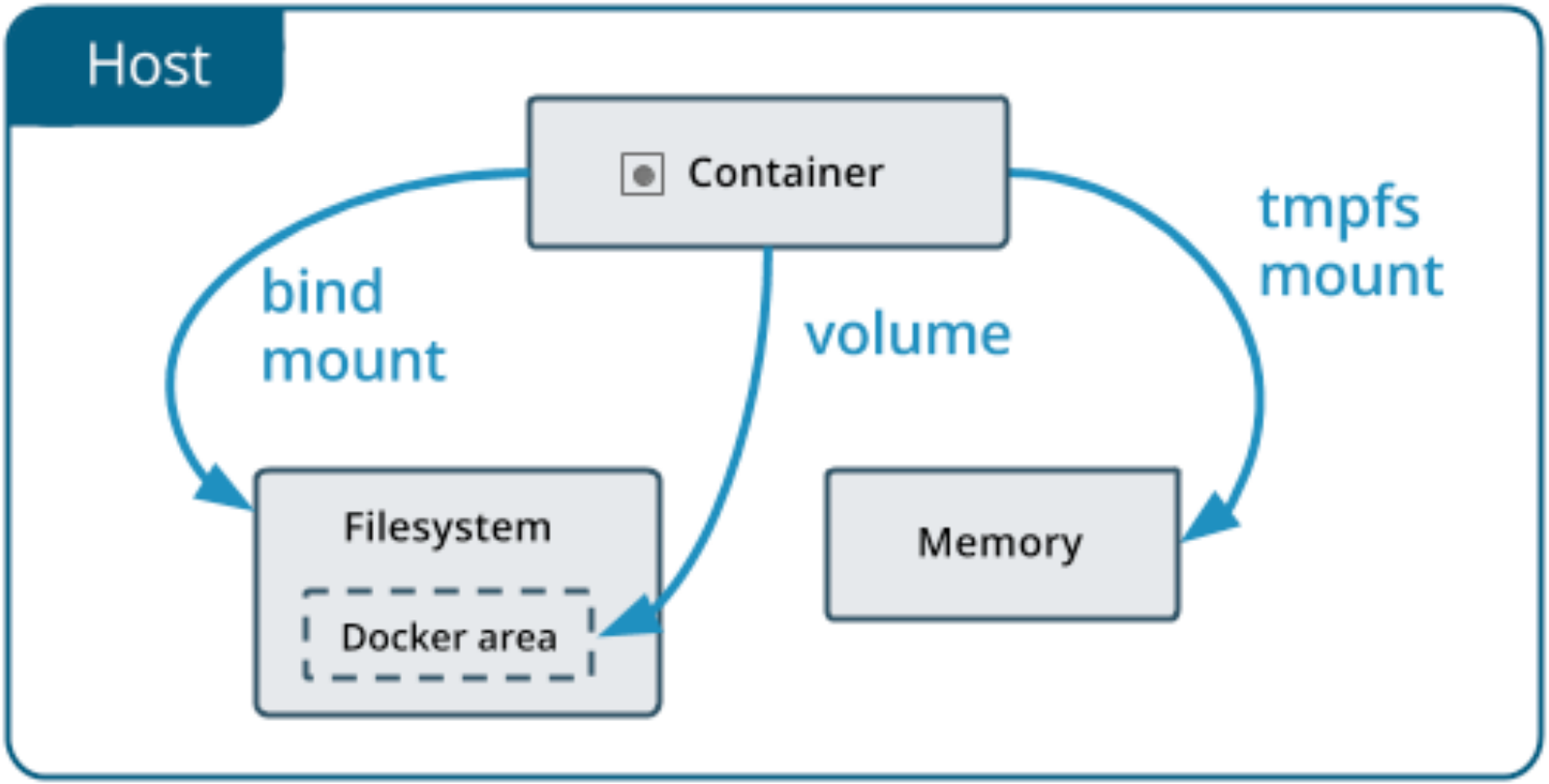
Docker has two options for containers to store files in the host machine, so that the files are persisted even after the container stops:

- ***Volumes***
- ***bind mounts***

If you're running Docker on Linux you can also use **a *tmpfs mount*.**

UNIVERSITAT DE BARCELONA

Manage data in Docker



UNIVERSITAT DE BARCELONA

Docker Volumes

- Created and managed by Docker. You can create a volume explicitly using the docker volume create command
- Volumes are stored within a directory on the Docker host.
- A given volume can be mounted into multiple containers simultaneously.
- When no running container is using a volume, the volume is still available to Docker and is not removed automatically. You can remove unused volumes using **docker volume prune.**
- Volumes also support the use of *volume drivers*, which allow you to store your data on remote hosts or cloud providers, among other possibilities.

UNIVERSITAT DE BARCELONA

Good use cases for volumes

- Sharing data among multiple running containers.
- When the Docker host is not guaranteed to have a given directory or file structure. Volumes help you decouple the configuration of the Docker host from the container runtime.
- When you want to store your container's data on a remote host or a cloud provider, rather than locally.
- When you need to back up, restore, or migrate data from one Docker host to another, volumes are a better choice. You can stop containers using the volume, then back up the volume's directory (such as /var/lib/docker/volumes/<volume-name>).

UNIVERSITAT DE BARCELONA

Bind Mounts

- Bind mounts have limited functionality compared to volumes.
- When you use a bind mount, a file or directory on the *host machine* is mounted into a container.
- The file or directory is referenced by its full path on the host machine.
- The file or directory does not need to exist on the Docker host already. It is created on demand if it does not yet exist.
- Bind mounts are very performant, but they rely on the host machine's filesystem having a specific directory structure available
- You can't use Docker CLI commands to directly manage bind mounts.

Good use cases for bind mounts

In general, you should use volumes where possible. Bind mounts are appropriate for the following types of use case:

- Sharing configuration files from the host machine to containers.
  - For example: This is how Docker provides DNS resolution to containers by default, by mounting /etc/resolv.conf from the host machine into each container.
- Sharing source code or build artifacts between a development environment on the Docker host and a container.
  - For instance, you may mount a Maven target/ directory into a container, and each time you build the Maven project on the Docker host, the container gets access to the rebuilt artifacts.
- When the file or directory structure of the Docker host is guaranteed to be consistent with the bind mounts the containers require.

UNIVERSITAT DE BARCELONA

Till Now

- What is Docker
- How to run Docker Containers from a given Image
- Next: Create Image and Run Multiple containers talking to each other.

UNIVERSITAT DE BARCELONA

Docker Images

Repository: A Docker Image repository is a place where Docker Images are actually stored.

Tag: The TAG refers to a particular snapshot of the image

Image ID: the IMAGE ID is the corresponding unique identifier for that image.

For simplicity, you can think of an image akin to a git repository - images can be committed with changes and have multiple versions. If you don't provide a specific version number, the client defaults to latest.

For example, you can pull a specific version of ubuntu image

$ docker pull ubuntu:18.04

UNIVERSITAT DE BARCELONA

Docker Images

Then there are official and user images, which can be both **base** and **child** images.

- **Official images** are images that are officially maintained and supported by the folks at Docker. These are typically one word long. In the list of images above, the python, ubuntu, busybox and hello-world images are official images.
- **User images** are images created and shared by users like you and me. They build on base images and add additional functionality. Typically, these are formatted as user/image-name.

UNIVERSITAT DE BARCELONA

Docker Images

An important distinction to be aware of when it comes to images is the difference between base and child images.

- **Base images** are images that have no parent image, usually images with an OS like ubuntu, busybox or debian.
- **Child images** are images that build on base images and add additional functionality.

Quick Quiz: prakhar1989/static-site is oficial image or user image? Is it Base image or Child image?

UNIVERSITAT DE BARCELONA

DockerFile

A Dockerfile is a simple text file that contains a list of commands that the Docker client calls while creating an image. It's a simple way to automate the image creation process. The best part is that the commands you write in a Dockerfile are almost identical to their equivalent Linux commands. This means you don't really have to learn new syntax to create your own dockerfiles.

Complete list of commands

https://docs.docker.com/engine/reference/builder/

UNIVERSITAT DE BARCELONA

Docker Hands On

$ git clone https://github.com/rohit-nlp/docker-curriculum

$ cd docker-curriculum/flask-app

Open the file DockerFile

Note: Remember this is the convención to name the file as DockerFile you can though name it anything and also to keep it in the root folder of the application.

UNIVERSITAT DE BARCELONA

Docker Hands On

We start with specifying our base image. Use the FROM keyword to do that

FROM python:3-onbuild

Since the application is written in Python, the base image being used is Python 3. More specifically, on-build version of the python image.

These images include multiple ONBUILD triggers, which should be all you need to bootstrap most applications. The build will COPY a requirements.txt file, RUN pip install on said file, and then copy the current directory into /usr/src/app

UNIVERSITAT DE BARCELONA

Docker Hands On

The next step usually is to write the commands of copying the files and installing the dependencies. Luckily for us, the onbuild version of the image takes care of that.

So the flask app uses port 5000 you need to expose it.

EXPOSE 5000

Note: Only exposed ports will be available from the container.

UNIVERSITAT DE BARCELONA

Docker Hands On

The last step is to write the command for running the application, which is simply - python ./app.py. Use the CMD command to do that -

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

The primary purpose of CMD is to tell the container which command it should run when it is started.

UNIVERSITAT DE BARCELONA

Docker Hands On

Now the DockerFile is ready will use it to build the image using build command.

$ docker build -t rohit1308k/catnip .

Before you run the command yourself (don't forget the period), make sure to replace my username with yours.

The docker build command is quite simple, it takes an optional tag name with -t and a location of the directory containing the Dockerfile.

$ docker run -p 8888:5000 rohit1308k/catnip

UNIVERSITAT DE BARCELONA

Docker Hands On

Lets publish our image

$ docker login

Or login from the docker desktop app.

Then push the image

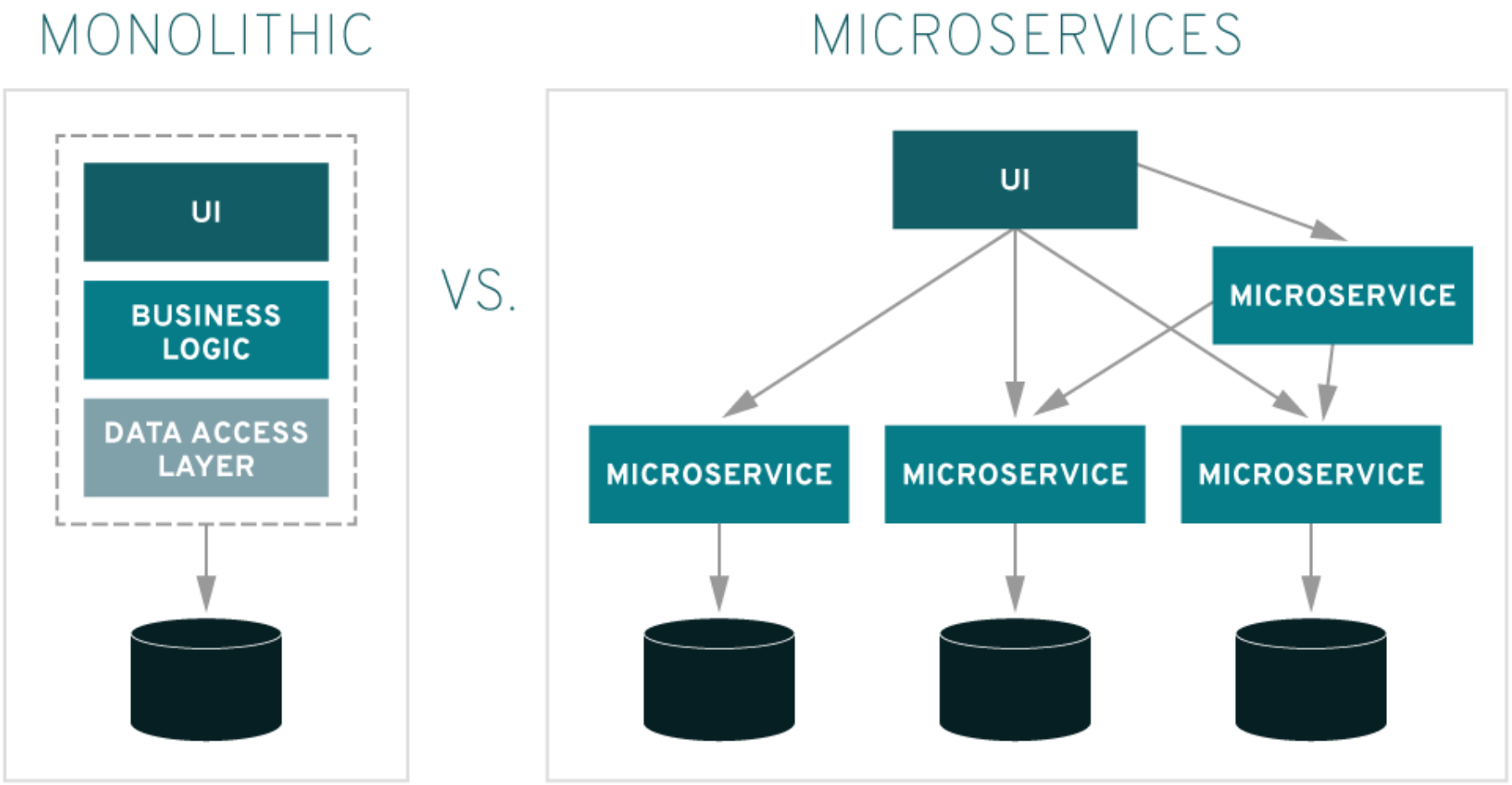
$ docker push rohit1308k/catnip

UNIVERSITAT DE BARCELONA

Multi Container Orchestration

UNIVERSITAT DE BARCELONA

Microservices



UNIVERSITAT DE BARCELONA

Learn by doing !!

So we are going to use Docker Compose to manage multi container environment

We will learn it by building a small app which has frontend in python and backend in elasticsearch.

SF Food Trucks: This is a simple app to search food trucks in San Francisco.

The data for the food trucks is made available in the public domain by SF Data.

And the simple code from*:
https://github.com/rohit-nlp/FoodTruckExample

ref*: https://github.com/prakhar1989/FoodTrucks

UNIVERSITAT DE BARCELONA

Hands on

Clone the simple application locally

$ git clone https://github.com/rohit-nlp/FoodTruckExample

Go inside the Project folder

$ cd FoodTrucksExample

List the files

$ tree -L 2

UNIVERSITAT DE BARCELONA

Hands on

The flask-app folder contains the Python application

The utils folder has some utilities to load the data into Elasticsearch.

The application consists of a Flask backend server and an Elasticsearch service.

That way if the app becomes popular, we can scale it by adding more containers depending on where the bottleneck lies.

UNIVERSITAT DE BARCELONA

Hands on

Now in previous example we already show how to build a flask web app container lets dig into Elasticsearch container !!

$ docker search elasticsearch

Note: Elastic, the company behind Elasticsearch, maintains its own registry for Elastic products. It's recommended to use the images from that registry if you plan to use Elasticsearch

$ docker pull docker.elastic.co/elasticsearch/elasticsearch:6.3.2

UNIVERSITAT DE BARCELONA

Hands on

Lets run elastic search

$ docker run -d --name es -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" docker.elastic.co/elasticsearch/elasticsearch:6.3.2

-e "discovery.type=single-node” is passing an enviornment variable to the container which will be used by the container to setup a single node cluster.

UNIVERSITAT DE BARCELONA

Hands on

Check logs on docker!!

$ docker container logs es

Quiz: Why es?? What else can we use?

Lets check our elastic serach node

$ curl 0.0.0.0:9200

UNIVERSITAT DE BARCELONA

Hands on

Now lets start our flask app

As we need more than just python3 for this app we will start with base image ubuntu.

Note: if you find that an existing image doesn't cater to your needs, feel free to start from another base image and tweak it yourself. For most of the images on Docker Hub, you should be able to find the corresponding Dockerfile on Github. Reading through existing Dockerfiles is one of the best ways to learn how to roll your own.

UNIVERSITAT DE BARCELONA

Hands on

FROM ubuntu:latest

We start off with the Ubuntu LTS base image and use the package manager apt-get to install the dependencies namely - Python and Node.

RUN apt-get -yqq update

RUN apt-get -yqq install python-pip python-dev curl gnupg

RUN curl -sL https://deb.nodesource.com/setup_10.x | bash

RUN apt-get install -yq nodejs

We install python and nodejs.

The yqq flag is used to suppress output and assumes "Yes" to all prompts.

UNIVERSITAT DE BARCELONA

Hands on

ADD flask-app /opt/flask-app

We then use the ADD command to copy our application into a new volume in the container - /opt/flask-app.

WORKDIR /opt/flask-app

We also set this as our working directory, so that the following commands will be run in the context of this location.

UNIVERSITAT DE BARCELONA

Hands on

RUN npm install

RUN npm run build

we configure Node by installing the packages from npm and running the build command as defined in our package.json file. This is specific to Node js

RUN pip install -r requirements.txt

Install the Python packages needed for our projet

UNIVERSITAT DE BARCELONA

Hands on

Lets build the image

$ docker build -t rohit1308k/foodtrucks-web .

Now lets run the image!!

$ docker run -P --rm rohit1308k/foodtrucks-web

Our flask app was unable to run since it was unable to connect to Elasticsearch. How do we tell one container about the other container and get them to talk to each other?

UNIVERSITAT DE BARCELONA

Hands on

$ docker container ls

So we have one ES container running on 0.0.0.0:9200 port which we can directly access.

If we can tell our Flask app to connect to this URL, it should be able to connect and talk to ES, right?

Lets check the code app.py

UNIVERSITAT DE BARCELONA

Hands on

To make this work, we need to tell the Flask container that the ES container is running on 0.0.0.0 host (the port by default is 9200) and that should make it work, right?

Unfortunately, that is not correct since the IP 0.0.0.0 is the IP to access ES container from the **host machine** i.e. from my Mac. Another container will not be able to access this on the same IP address.

UNIVERSITAT DE BARCELONA

Docker Network

When docker is installed, it creates three networks automatically.

```
NETWORK ID          NAME                DRIVER              SCOPE
c2c695315b3a        bridge              bridge              local
a875bec5d6fd        host                host                local
ead0e804a67b        none                null                local
```

Lets have quick look.

UNIVERSITAT DE BARCELONA

Docker Network

Bridge: The default network driver. If you don't specify a driver, this is the type of network you are creating. **Bridge networks are usually used when your applications run in standalone containers that need to communicate.** So that means that when I ran the ES container, it was running in this bridge network.

host: For standalone containers, remove network isolation between the container and the Docker host, and use the host's networking directly. host is only available for swarm services on Docker 17.06 and higher

none: For this container, disable all networking. Usually used in conjunction with a custom network driver. none is not available for swarm services

UNIVERSITAT DE BARCELONA

Docker Network

- Lets check the network

$ docker network inspect bridge

You can see that our container **es** is listed under the Containers section in the output. What we also see is the IP address this container has been allotted - 172.17.0.3.

Is this the IP address that we're looking for? Let's find out by running our flask container and trying to access this IP.

UNIVERSITAT DE BARCELONA

Hands On

$ docker run -it --rm rohit1308k/foodtrucks-web bash

$ curl 172.17.0.3:9200

Although we have figured out a way to make the containers talk to each other, there are still two problems with this approach -

- How do we tell the Flask container that es hostname stands for 172.17.0.2 or some other IP since the IP can change?
- Since the *bridge* network is shared by every container by default, this method is **not secure**. How do we isolate our network?

UNIVERSITAT DE BARCELONA

Hands On

The good news that Docker has a great answer to our questions. It allows us to define our own networks while keeping them isolated using the docker network command.

Let's first go ahead and create our own network.

$ docker network create foodtrucks-net

Lets view the networks again

$ docker network ls

UNIVERSITAT DE BARCELONA

Hands On

The network create command creates a new ***bridge*** network, which is what we need at the moment.

In terms of Docker, a bridge network uses a software bridge which allows containers connected to the same bridge network to communicate, while providing isolation from containers which are not connected to that bridge network. The Docker bridge driver automatically installs rules in the host machine so that containers on different bridge networks cannot communicate directly with each other.

There are other kinds of networks that you can create, and you are encouraged to read about them in the official docs.

https://docs.docker.com/network/

UNIVERSITAT DE BARCELONA

Hands On

Now that we have a network, we can launch our containers inside this network using the --net flag.

First lets clean up

$ docker container stop es

$ docker container rm es

Now run both the container on the same network.

$ docker run -d --name es --net foodtrucks-net -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" docker.elastic.co/elasticsearch/elasticsearch:6.3.2

$ docker run -it --rm --net foodtrucks-net rohit1308k/foodtrucks-web bash

$curl es:9200 (from inside the container)

$ docker run -d --net foodtrucks-net -p 5000:5000 --name foodtrucks-web rohit1308k/foodtrucks-web

UNIVERSITAT DE BARCELONA

Hands on

In summary what we did are:

build the flask container

docker build -t rohit1308k/foodtrucks-web .

create the network

docker network create foodtrucks-net

start the ES container

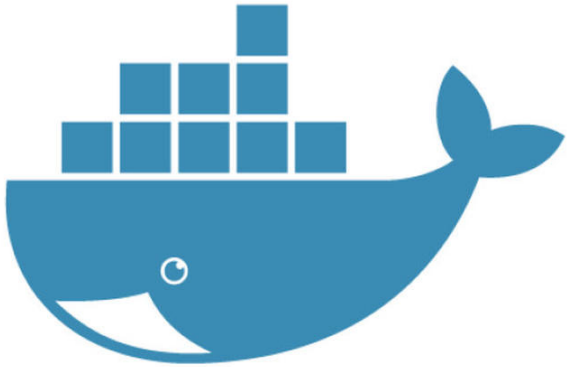
docker run -d --name es --net foodtrucks-net -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" docker.elastic.co/elasticsearch/elasticsearch:6.3.2

start the flask app container

docker run -d --net foodtrucks-net -p 5000:5000 --name foodtrucks-web rohit1308k/foodtrucks-web

UNIVERSITAT DE BARCELONA

Multi Container



Web App

Database

Messaging Service

Reporting Service

Email Service

How to manage all these different containers?

UNIVERSITAT DE BARCELONA

Docker Compose

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

Using Compose is basically a three-step process:

1. Define your app's environment with a Dockerfile so it can be reproduced anywhere.
2. Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
3. Run docker-compose up and Compose starts and runs your entire app.

UNIVERSITAT DE BARCELONA

Hands on

Lets everything we did before using Docker Compose

1. Test the installation (it comes by default when you install in mac or windows from docker tool box)

$ docker-compose –versión

Now open the file docker-compose.yml in the root directory.

UNIVERSITAT DE BARCELONA

Docker Compose

We define 2 important things

1. Services
2. Volumes (optional but mostly always used)

At the parent level, we define the names of our services: es and web.

For each service that Docker needs to run, we can add additional parameters out of which image is required.

- For es, we just refer to the elasticsearch image available on Elastic registry.
- For our Flask app, we refer to the image that we built at the beginning of this section.

UNIVERSITAT DE BARCELONA

Docker Compose

Via other parameters such as **command** and **ports** we provide more information about the container.

The **volumes** parameter specifies a mount point in our web container where the code will reside.

UNIVERSITAT DE BARCELONA

Docker Compose

Lets first stop and clean old runs

$ docker stop es foodtrucks-web

$ docker rm es foodtrucks-web

Now lets see

$ docker-compose up

Thats all !! docker-compose up looks up a docker-compose.yml file in its current folder

UNIVERSITAT DE BARCELONA

Hands On

To stop the services

Try

$ docker-compose down
To stop the containers

Or

$ docker-compose down –v

To Stop the containers and delete the volumes

also remove the foodtrucks network that we created last time.

$ docker network rm foodtrucks-net

UNIVERSITAT DE BARCELONA

Hands on

Finally lets run in detached mode

$ docker-compose up –d

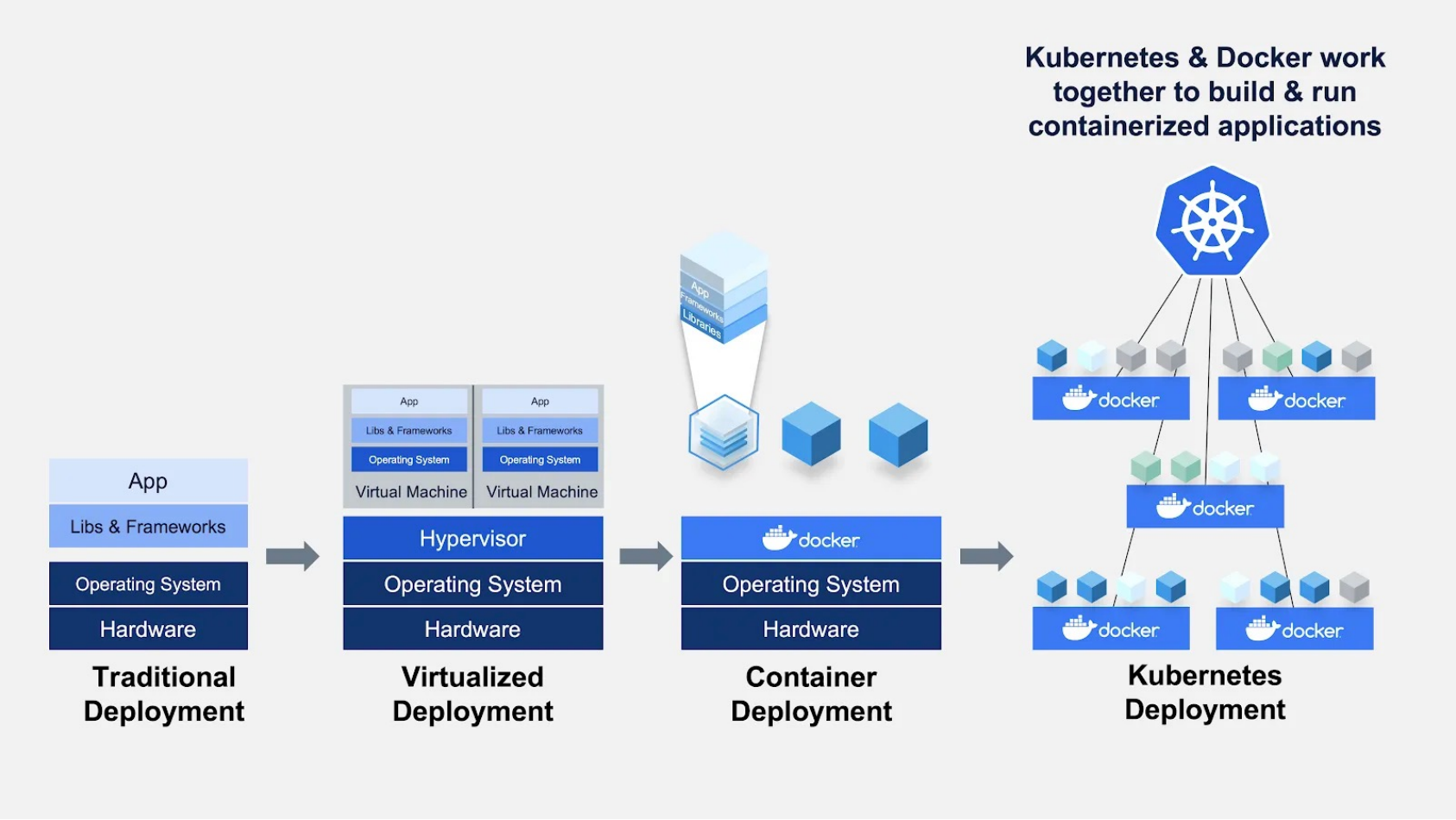
Lets see what happened in the network

$ docker network ls

You can see that compose went ahead and created a new network called foodtrucks_default and attached both the new services in that network so that each of these are discoverable to the other

UNIVERSITAT DE BARCELONA

Docker orchestration



UNIVERSITAT DE BARCELONA

Try at home

Kubernetes

- https://kubernetes.io/docs/tutorials/
- https://kubernetes.io/docs/tutorials/kubernetes-basics/

UNIVERSITAT DE BARCELONA

Thank you!