# CURSOR Python / Docker Lesson

## Lesson Plan

1. 😏 Containerization vs virtualization?
2. 😟 What is Docker?
3. 🤠 Terminology
4. 😳 Setup Docker locally!
5. 😁 Writing first Dockerfile!
6. 😷 Writing first docker-compose.yml
7. 😤 Constraints
8. 😌 Docker Cloud
9. 🤑 Deploy to AWS
10. 🙄 Docker commands
11. 🙄 Kubernetes
12. Recommend 2 articles from @Medium
13. Recommend Roman's slideshare

---

## 😈 Who am I?

Ask me :)

## 😌 What is Docker?

VMs (Virtual Machines) run applications

inside a guest Operating System
VMs are great at providing full process isolation for applications.

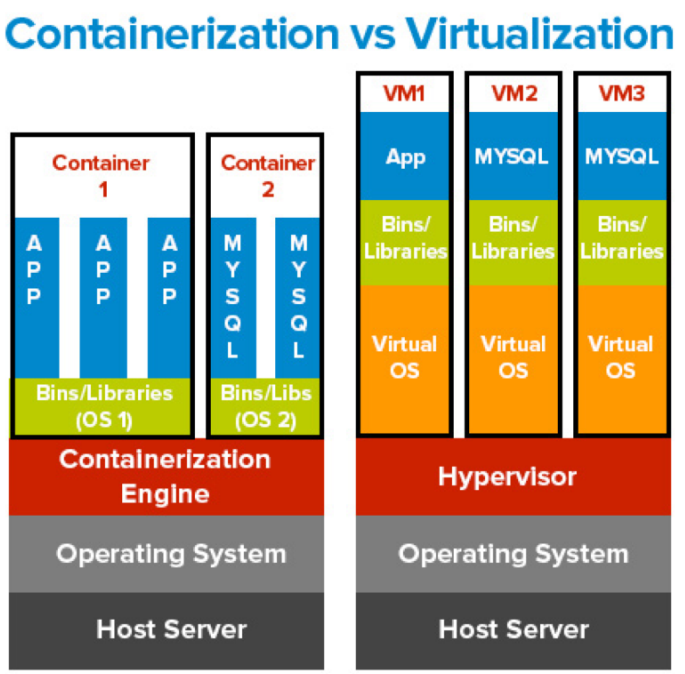
Vagrant

## Docker is a tool, that allows developers to easily deploy their applications in a sandbox (containers).

Docker is a platform for packaging, deploying, and running applications.

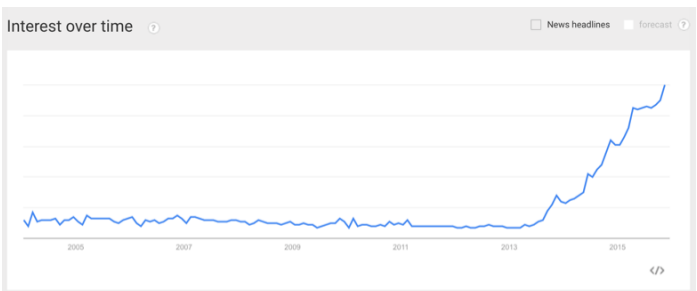
**What are containers?**

## MacOS = kernel v4.5 + fs1 + tools1 + ui1
## Ubuntu = kernel v4.5 + fs2 + tools2 + ui2




Docker applications run in containers that can be used on any system: a developer's laptop, systems on premises, or in the cloud.

Containers use less memory and less CPU.




## Benefits

- faster development process
- handy application encapsulation
- the same behaviour on local machine / dev / staging / production servers
- easy and clear monitoring
- easy to scale

## 😨 Terminology

- **Image** - the basic element for every container.
- **Container** - a running instance that encapsulates the required software.

Instructions -> build to Image -> container
i.e.: instructions "setup python 2.7" + "setup Flask" + "configure Flash" >> image >> container1, container2, ...

- **Port** - a TCP/UDP port in its original meaning.
- **Volume** - can be described as a shared folder.

- **Registry** - the server that stores Docker images.
- **DockerHub** - a registry with the web interface provided by Docker Inc.

## simple flow:

developer: (1) build image, (2) push image to remote Docker repo
staging-server: (3) pull new image, (4) drop old container, (5) run new container

**complex flow:**
developer: (1) push changes to Git repo
CI-server: (2) monitoring changes, (3) build image on new changes, (4) push image to remote Docker repo
testing-server: (5) monitoring for new images, (6) if new images appers, will pull new image, and (7) start container
production-server: (8) ..but for master branch only

## 😏 Setup Docker locally!

https://www.docker.com/
https://docs.docker.com/

$ docker run hello-world

$ docker run ubuntu /bin/echo 'Hello

world'
$ docker run -i -t --rm ubuntu /bin/bash

## 😷 BusyBox

$ docker pull busybox
$ docker images
$ docker run busybox
$ docker run busybox echo "Hello!"
$ docker ps
$ docker ps -a
$ docker run -it busybox sh
$ docker rm

## 😳 Running over python

$ docker rm

## 😳 Writing first Dockerfile

**FROM** python:latest
**ENV** NAME World

**WORKDIR** /app
**ADD** . /app

**RUN** pip install -U pip
**RUN** pip install -U requirements.txt
**RUN** pip install --trusted-host pypi.python.org Flask

**EXPOSE** 5000

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

app.py

```
from flask import Flask
import os
import socket

app = Flask(__name__)
@app.route("/")

def hello():
    html = "<h3>Hello {name}!</h3> <b>Hostname:</b> {hostname}<br/>"
    return html.format(name=os.getenv("NAME", "world"), hostname=socket.gethostname())
```

```
    if __name__ == "__main__":
        app.run(host='0.0.0.0',
    port=4000)
```

## 🤠 docker-compose.yml

```
version: '2'
services:
  web:
    image: django
    build: .
    volumes:
      - .:/code
    environment:
      IN_DOCKER: 1
```

```
$ docker-compose up -d
```

## 😤 Constraints

- **1 application = 1 container**.
- Run the process in the **foreground**.
- **Keep data out of containers** — use volumes.
- **Do not use SSH.** If you need to step into a container, you can use the docker exec command.

- **Avoid manual configurations** or actions inside containers.

## 😤 Docker Cloud

https://cloud.docker.com/

$ docker login
$ docker tag mypyweb ericgoebelbecker/stackify-tutorial:1.00
$ docker push ericgoebelbecker/stackify-tutorial:1.00

http://hub.docker.com

$ docker run -p 8080:4000 --name webapp -e NAME="Docker Hub" ericgoebelbecker/stackify-tutorial:1.00

## 🤑 Deploy to AWS

https://aws.amazon.com/
https://hub.docker.com/

https://docker-curriculum.com/#docker-on-aws

## 🙄 Docker commands

List of commands – $ docker ps -a

List of images – $ docker images

docker build -t mynginx .

docker run --name foo -v /souce:/dest:ro -p 8080:80 -d nginx

docker stop {container}
docker rm {container}

$ docker port {container}

---

## Homework

- https://gist.github.com/itspoma/ce1337edd6d83736b24c48978d671027
- подивитись https://www.accelebrate.com/blog/javascript-es6-classes-and-prototype-inheritance-part-1-of-2/
- подивитись https://www.youtube.com/watch?v=QyUFheng6J0
- подивитись https://github.com/getify/You-Dont-Know-JS/tree/master/this%20%26%20object%20prototypes

--

Cheers!

https://cursor.education/teacher/roman-rodomansky