Tema 2

Docker Compose

Universidad
Rey Juan Carlos

Docker Compose

- Es una herramienta para definir aplicaciones formadas por **varios contenedores**
- Un fichero YAML define los **contenedores** (imagen, puertos, volúmenes…) y cómo se **relacionan** entre sí
- Los contenedores se comunican:
  - Protocoles de red
  - Volúmenes compartidos

https://docs.docker.com/compose/

Docker Compose

- La herramienta docker-compose tiene que **instalarse** por separado en linux (no viene incluida con docker)
- El fichero YAML se suele llamar

`docker-compose.yml`

- En la carpeta donde está el fichero, la aplicación se ejecuta con el comando

```
$ docker-compose up
```

https://docs.docker.com/compose/install/

Docker Compose

- **Definición de cada contenedor**
  - Imagen
    - Puede descargarse de DockerHub
    - Puede estar construida localmente
    - Puede construirse con un Dockerfile en el momento de iniciar la aplicación

Docker Compose

- **Definición de cada contenedor**
  - **Puertos:**
    - Mapeados en el host (para ser usados desde localhost)
    - No mapeados (sólo se pueden conectar otros contenedores)
  - **Volúmenes:**
    - Carpetas del host accesible desde el contenedor
    - Compartidos entre contenedores (en una carpeta interna de docker)

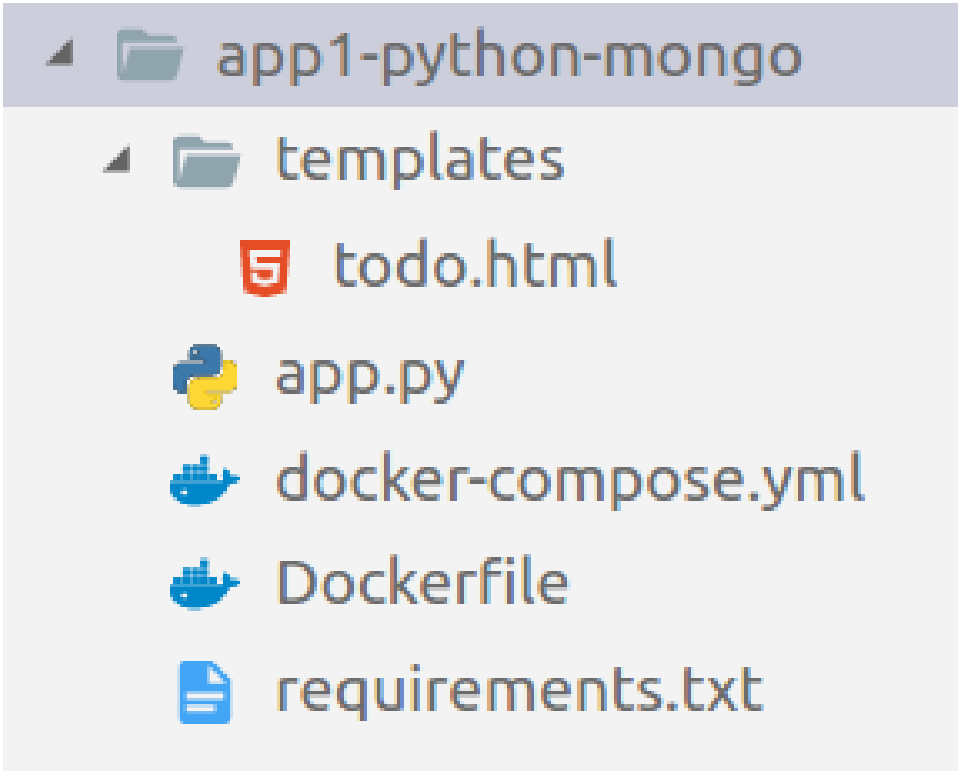
Docker Compose

- **Ejemplo: App web con BBDD**
  - Web con tecnología Python y framework Flask
  - BBDD MongoDB
  - 2 Contenedores

http://containertutorials.com/docker-compose/flask-mongo-compose.html

Docker Compose

- **Ejemplo: App web con BBDD**



https://github.com/codeurjc/curso-docker/tree/master/app1-python-mongo

Docker Compose

- Python web app

app.py

```
import os
from flask import Flask, redirect, url_for, request, render_template
from pymongo import MongoClient

app = Flask(__name__)

client = MongoClient('db', 27017)
db = client.tododb


@app.route('/')
def todo():

    _items = db.tododb.find()
    items = [item for item in _items]

    return render_template('todo.html', items=items)


@app.route('/new', methods=['POST'])
def new():

    item_doc = {
        'name': request.form['name'],
        'description': request.form['description']
    }
    db.tododb.insert_one(item_doc)

    return redirect(url_for('todo'))

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

Docker Compose

- **HTML Template**

templates/todo.html

```
<form action="/new" method="POST">
  <input type="text" name="name"></input>
  <input type="text" name="description"></input>
  <input type="submit"></input>
</form>

{% for item in items %}
  <h1> {{ item.name }} </h1>
  <p> {{ item.description }} </p>
{% endfor %}
```

- **App libraries**

requirements.txt

```
flask
pymongo
```

Docker Compose

- **Dockerfile de la aplicación web**

Dockerfile

```
FROM python:2.7
ADD . /todo
WORKDIR /todo
RUN pip install -r requirements.txt
CMD ["python", "-u", "app.py"]
```

- Podemos construir la imagen con este Dockerfile desde línea de comandos si queremos, pero vamos a usar docker-compose para que lo haga

Docker Compose

- Fichero docker-compose.yml

Servicios

Cada contenedor es un servicio de la app

La propiedad indica el nombre del servicio

Esta app tiene dos servicios: web y db

docker-compose.yml

```
web:
  build: .
  ports:
    - "5000:5000"
  volumes:
    - .:/todo
  links:
    - db
db:
  image: mongo:3.0.2
```

Docker Compose

- Fichero docker-compose.yml

build

Se indica la ruta del Dockerfile para construir la imagen

docker-compose.yml

```
web:
  build: .
  ports:
    - "5000:5000"
  volumes:
    - .:/todo
  links:
    - db
db:
  image: mongo:3.0.2
```

Docker Compose

- Fichero docker-compose.yml

image

Se indica el nombre de la imagen en Dockerhub o en local

docker-compose.yml

```
web:
  build: .
  ports:
    - "5000:5000"
  volumes:
    - .:/todo
  links:
    - db
db:
  image: mongo:3.0.2
```

Docker Compose

- Fichero docker-compose.yml

docker-compose.yml

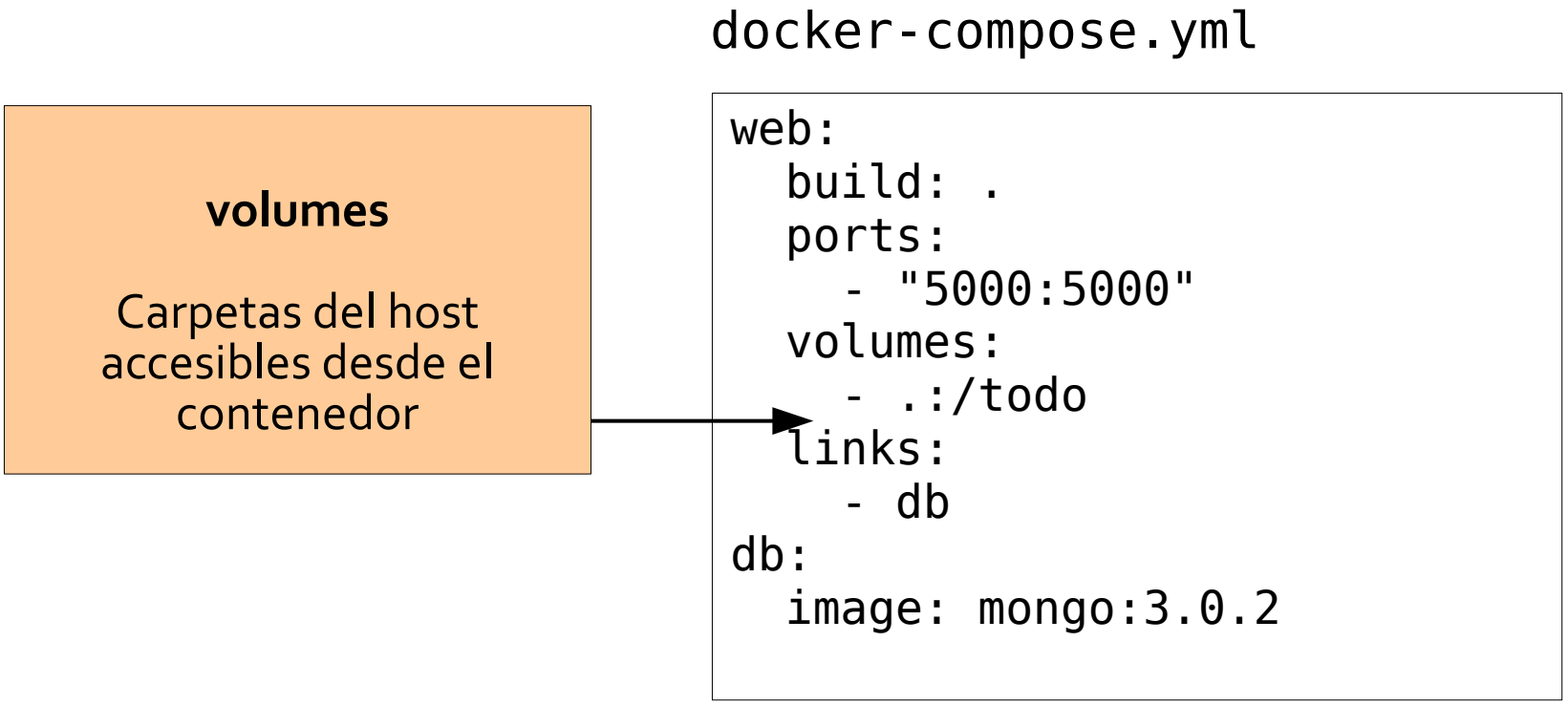
ports

Puertos "bindeados" al host. Como la opción -p al arrancar un contenedor

```
web:
  build: .
  ports:
    - "5000:5000"
  volumes:
    - .:/todo
  links:
    - db
db:
  image: mongo:3.0.2
```

Docker Compose

- Fichero docker-compose.yml



Docker Compose

- Fichero docker-compose.yml

links

Permite acceder al servicio 'db' desde el servicio 'web' usando como nombre del host 'db'

docker-compose.yml

```
web:
  build: .
  ports:
    - "5000:5000"
  volumes:
    - .:/todo
  links:
    - db
db:
  image: mongo:3.0.2
```

Docker Compose

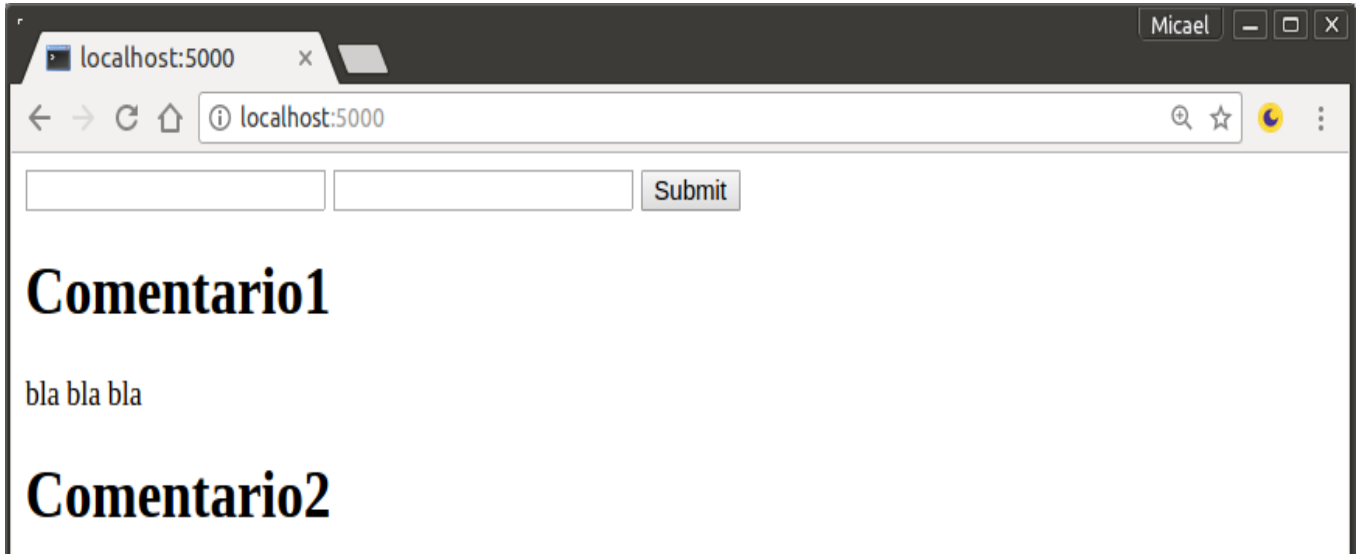
- **Arrancar la aplicación**

```
$ docker-compose up
```

  - Construye la imagen si no está construida ya.
    - Si la imagen está disponible, no se reconstruye aunque cambie el código o el Dockerfile, es necesario ejecutar **docker-compose up --build**
  - Se descarga la imagen de MongoDB si no está en local
  - Inicia los dos contenedores

Docker Compose

- **Arrancar la aplicación**
  - La app está disponible en http://locahost:5000/



Nota: Con Docker Toolbox la web está disponible en la IP de la VM (obtenida con el comando **docker-machine ip default**)

Docker Compose

- **¿Cómo funciona?**
  - Se muestran los logs de todos los contendores
  - Para parar la app, **Ctrl+C** en la consola (también **docker-compose stop**)
  - Si se para y arranca de nuevo el servicio sin cambiar la configuración de un contenedor, **se vuelve a iniciar el mismo** (no se crea uno nuevo). Los datos de la BBDD no se pierden porque están dentro del contendor

Docker Compose

- **Ideal para desarrollo**
  - Podemos definir una app con múltiples contenedores en un fichero de texto (y subirlo a un repositorio)
  - Cualquier desarrollador puede arrancar la app sin tener nada instalado en local (sólo docker y docker-compose)
  - Es muy cómodo iniciar y parar todos los servicios a la vez y sólo cuando realmente se necesitan (no tienen que estar iniciados al arrancar la app)
  - Todos los logs centralizados

Docker Compose

- **Distribución de apps dockerizadas**
  - Si todos los contenedores del compose están en DockerHub, para distribuir una app multicontenedor dockerizada basta con descargar el docker-compose.yml y arrancarlo.
  - Con curl disponible y el docker-compose.yml en github:

```
curl https://raw.githubusercontent.com/phundament/app-tutum/docker-compose.yml \
    | docker-compose -f - up -d
```

Ejercicio 7

- **Dockeriza la aplicación java-webapp-bbdd**
  - Enunciado **curso-docker/java-webapp-bbdd-enunciado**
  - Se usará docker-compose
  - La aplicación necesita una BBDD MySQL
    - Password de root: pass
  - En una aplicación SpringBoot se puede configurar la ruta de la BBDD y el esquema con la variable de entorno:
    - SPRING_DATASOURCE_URL=jdbc:mysql://<host>/<database>
  - Desactivar los tests al construir la app Maven: -DskipTests=true