
VENOM
BY @MANUFOSELA

V.E.No.M y Docker

El Stack para el Desarrollo Simbionte

Un poco de historia

En 2009 irrumpen nodejs y mongodb.

En 2010 aparecen AngulaJS y Express.

Poco después aparecen MEAN o METEOR.

Los stacks de la década pasada, LAMP y WAMP junto con ExtJS y jQuery, dan paso en esta década a los stacks de desarrollo javascript: MEAN, METEOR, MERN

Justo llegando al final, con la irrupción de VueJS aparece el stack VENoM

¿Preguntas?

- Qué es
- Para qué
- Por qué
- Qué se requiere
- Qué desventajas tiene
- Qué aporta en el mundo Cloud

¿Qué es?

V.E.No.M. es el acrónimo de **V**ueJS **E**xpress **No**deJS y **M**ongoDB

Es un stack completo de desarrollo de aplicaciones web.

- VueJS para el frontend
- NodeJS con Express para el backend / API
- MongoDB para la base de datos

Eso como base. Junto a ellos pueden aparecer otras muchas dependencias.

¿Para qué?

Para tener un entorno isomórfico de desarrollo javascript.

Para tener el control de lo que haces en cliente y en servidor.

Para tener el control de tus datos.

¿Por qué?

- Porque al tener un desarrollo isomórfico puedes reutilizar clases, objetos, funciones e incluso dependencias... en front y en back.
- Porque conforme tu aplicación crezca tendrás el control para implementar esas necesidades que van surgiendo inevitablemente.
- Porque la base de datos es tuya y tener los datos alojados en sistemas de terceros supone asumir sus normas.

¿Qué “desventajas” tiene?

- ⊗ Requiere de conocimientos de front y back.
- ⊗ Tienes que ser Juan Palomo.
- ⊗ Como al personaje del comic, te acaba poseyendo.

¿Podemos mejorar el stack?

Dockerizandolo, Porque así:

- Mantiene las diferentes partes aisladas.
- Entregas un desarrollo que garantizas que funciona.
- Es escalable.

Demo time

¿Qué vamos a hacer?

1. ¡Vamos a usar VENoM!.
2. scaffolding de un proyecto por defecto de Vue.
3. Un servidor de un API Rest sencillo.
4. Un servidor de mongoDB como base de datos.
5. Docker para separar los entornos de desarrollo, api y base de datos.
6. Docker-compose para orquestar todos los contenedores Docker.

Lo básico

1. Instalamos docker y docker-compose:
   ```
   # sudo apt install docker docker-compose
   ```
2. Añadimos al usuario de ubuntu al grupo docker y reiniciamos
   ```
   # sudo usermod -aG docker $USER
   ```
3. En https://hub.docker.com/ podemos buscar imagenes de docker ya preparadas
4. Podemos comprobar qué imágenes tenemos con:
   ```
   # docker images
   ```

Instalación de node y Vue

sudo apt install nodejs

o bien

curl -OJ

https://nodejs.org/dist/v10.13.0/node-v10.13.0-linux-x64.tar.xz

tar -xvf node-v10.13.0-linux-x64.tar.xz

sudo mv node-v10.13.0-linux-x64 /opt

sudo chown $USER.$USER /opt/node-v10.13.0-linux-x64

sudo ln -s /opt/node-v10.13.0-linux-x64 /opt/node

export PATH=$PATH:/opt/node/bin/:/opt/node/lib/node_modules/

npm install -g @vue/cli

vue -V

Creamos workspaces.
Instalamos Express

```
# mkdir proyecto
# cd proyecto
# mkdir api
# cd api
# npm init -y
# npm install --save express body-parser method-override
```

Creamos proyecto en Vue

```
# cd ..
# npm install -g @vue/cli
# vue -V
# vue create front -d
# cd front
# npm run serve
```

Creamos Dockerfile.front para vue

```
# cd ..


Creamos y editamos Dockerfile.front


 FROM node:8
 WORKDIR /usr/src/app
 EXPOSE 8080
 CMD [ "npm", "run", "serve" ]
```

Construimos y ejecutamos Docker para vue

Para construir la imagen de docker:

```
# docker build -f Dockerfile.front -t front-vue .
```

Para crear el contenedor a partir de esa imagen:

```
# docker run -d --name "front-vue-container" -p 8080:8080 -v
$(pwd)/front:/usr/src/app front-vue
```

Comprobamos:

```
# docker ps
# docker inspect front-vue
# docker logs front-vue
```

Creamos Dockerfile.front2 para vue

Podemos hacer que se instale Vue en el contenedor, para usar solamente las carpetas con el código. Creamos y editamos el fichero Dockerfile.front2:

```
FROM node:8
WORKDIR /usr/src/app
COPY front/package.json ./
COPY front/package-lock.json ./
RUN npm install
EXPOSE 8080
CMD [ "npm", "run", “serve” ]
```

Construimos y ejecutamos Docker para vue

Para construir la imagen de docker:

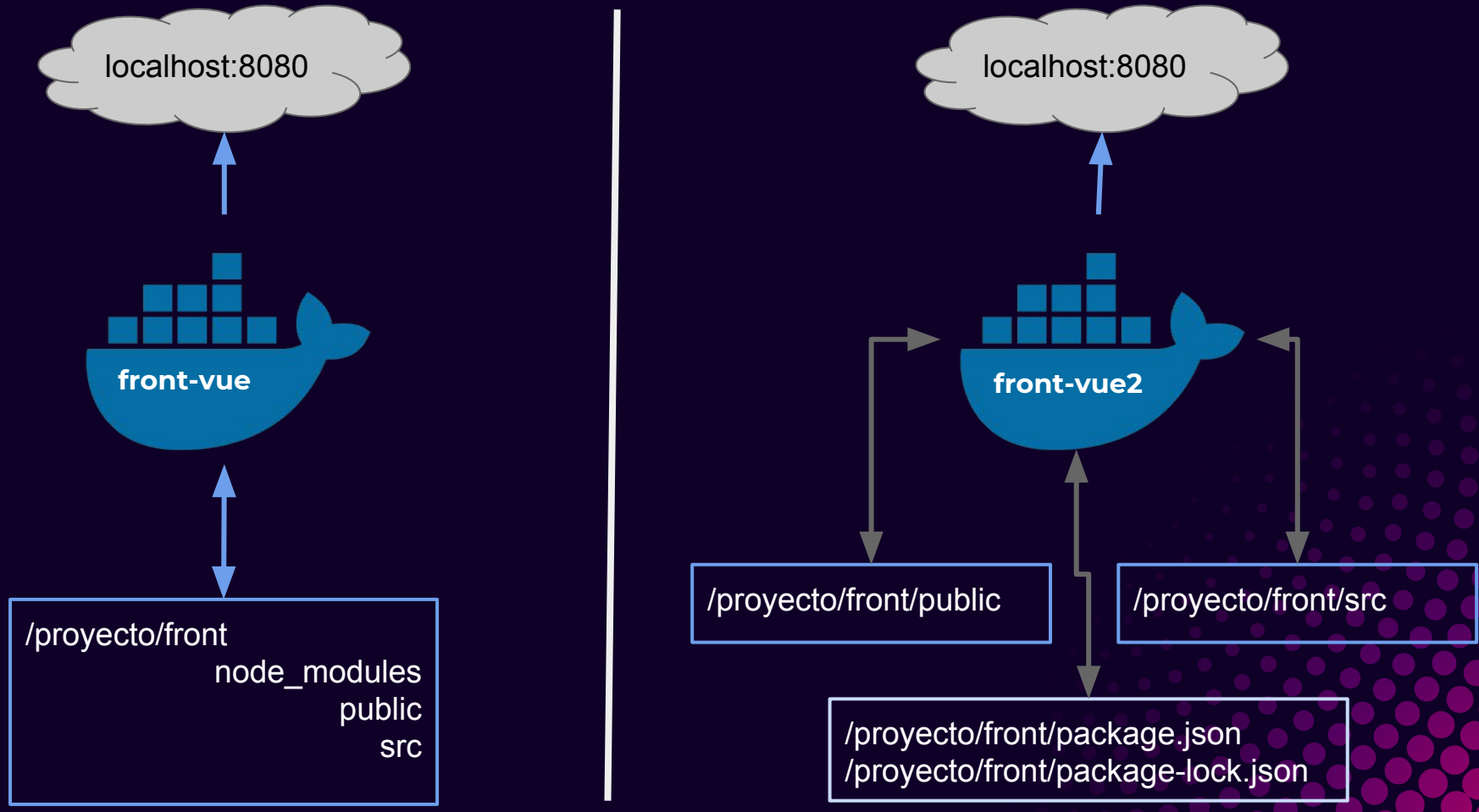
```
# docker build -f Dockerfile.front2 -t front-vue2 .
```

Para crear el contenedor a partir de esa imagen:

```
# docker run -d --name "front-vue-container2" -p 8080:8080 -v
$(pwd)/front/public:/usr/src/app/public -v
$(pwd)/front/src:/usr/src/app/src front-vue2
```

Comprobamos:

```
# docker ps
# docker inspect front-vue
# docker logs front-vue
```


localhost:8080
front-vue
/proyecto/front
node_modules
public
src
localhost:8080
front-vue2
/proyecto/front/public
/proyecto/front/src
/proyecto/front/package.json
/proyecto/front/package-lock.json

Construimos el api con node y express

```
const express = require('express');
const app = express();

const PORT = 3000;

app.get('/', function(req, res) {
  res.json({"hello": "express with mongo"});
});

app.get('/misdatos', function(req, res) {
     res.json({"misdatos": "son estos"});
});

app.listen(PORT, function() {
  console.log('Your node js server is running on PORT:', PORT);
});
```

Construimos el contenedor del api

Vamos a crear un nuevo contenedor con node y express.

Creamos el fichero Dockerfile.api:

```
FROM node:8
WORKDIR /usr/src/app
COPY api/package.json ./
COPY api/package-lock.json ./
COPY api/index.js ./
RUN npm install
EXPOSE 3000
CMD [ "node", "index.js" ]
```

Construimos el contenedor del api

Para construir la imagen de docker:

```
# docker build -f Dockerfile.api -t api-express .
```

Para crear el contenedor a partir de esa imagen:

```
# docker run -d --name "api-express-container" -p 3000:3000 api-express
```

Comprobamos:

```
# docker ps
# docker inspect api-express
# docker logs api-express
```

Construimos el contenedor de mongo

Para la parte de mongo, como no vamos a configurar nada en nuestro Dockerfile podemos hacer simplemente:

```
# docker pull mongo
```

Esto nos baja la imagen de mongodb que luego podremos usar para construir nuestro contenedor.

Construimos el contenedor de mongo

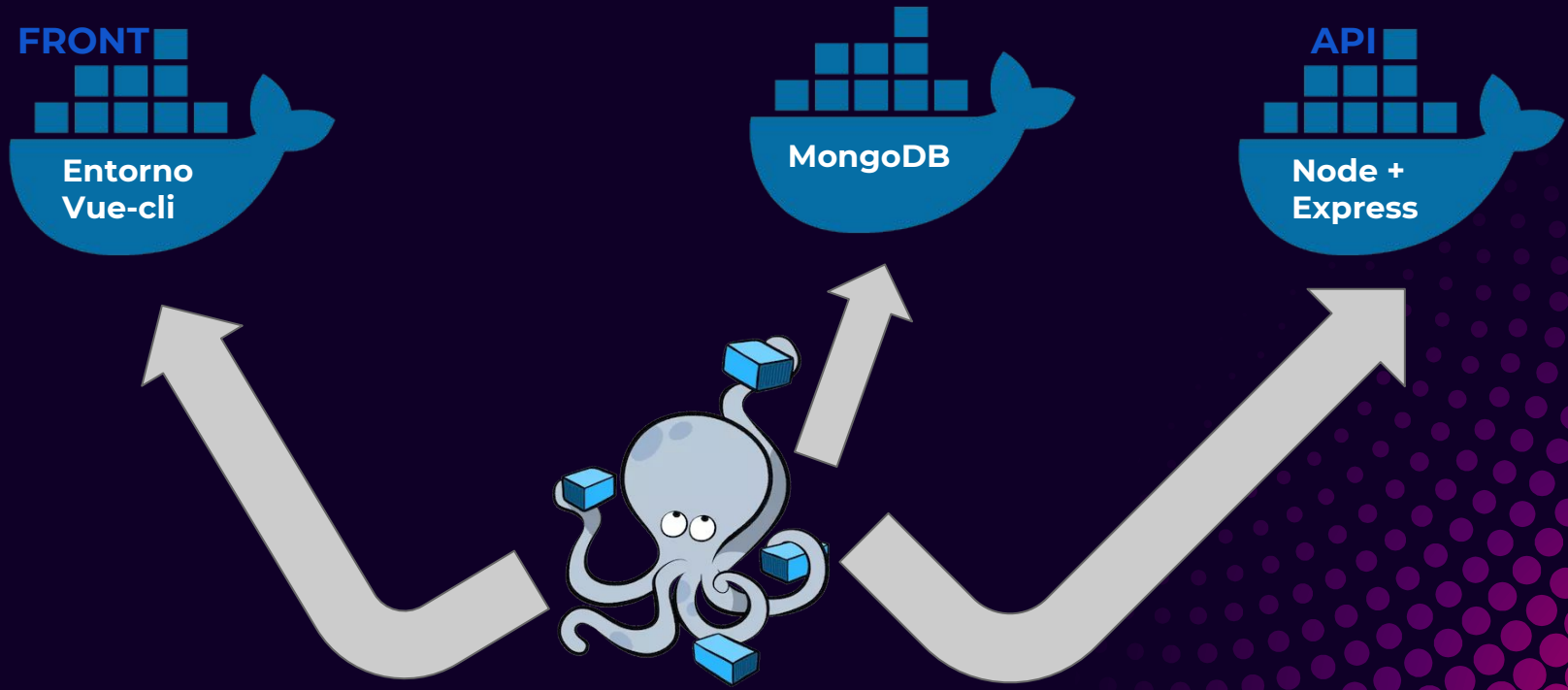
Para crear el contenedor a partir de esa imagen:

```
# docker run -d --name "mongo-container" -it mongo
```

Comprobamos:

```
# docker ps
# docker inspect mongo
# docker logs mongo
# mongo
>
```

Coordinamos todo con docker-compose



Configuración de Docker-compose

La configuración de docker-compose se hace mediante un fichero .yml

En este indicamos las build de los contenedores docker que vamos a usar y la relación entre ellos.

Docker-compose.yml

```yaml
version: "3"
services:
  front-vue:
    container_name: front-vue-container
    build:
      context: .
      dockerfile: Dockerfile.front2
    volumes:
      - ./front/public:/usr/src/app/public
      - ./front/src:/usr/src/app/src
    ports:
      - "8080:8080"
    links:
      - api-rest
  api-rest:
      container_name: api-rest-container
      restart: always
      build:
        context: .
        dockerfile: Dockerfile.api
      ports:
        - "3000:3000"
      links:
        - mongo
  mongo:
      container_name: mongo-container
      image: mongo
      volumes:
        - ./mongo.db:/data/db
      ports:
        - "27017:27017"
```

Construimos el contenedor del api y mongo

Lo primero nos aseguramos que no tenemos contenedores/imagenes creados.

Para generar los contenedores coordinados con docker-compose:

```
# docker-compose up -d
```

Comprobamos:

```
# docker-compose ps
```

Probamos a entrar en:

http://localhost:8080
http://localhost:3000
http://localhost:27017

Cuando terminemos nuestra app...

- Al terminar deberemos cambiar el dockerfile de VUE para que clone un repo y para que haga la build.
- Y en vez de usar el servidor de vue-cli lo ideal sería usar un servidor como NGINX
- Si usamos el servidor NGINX podemos usar este para que haga de proxy con nuestro servidor REST, de manera que todas las llamadas, por ejemplo a /api, las encamine hacia el puerto 3000.
- Lo que entregamos para subir a otros entornos pre-productivos es un repo que contiene los ficheros Dockerfile y docker-compones.yml
- Con esto garantizamos que siempre, siempre, va a funcionar en cualquier lado.

Se termino la frase “En mi local funciona"

Si funciona en tu Docker, siempre funciona

BONUS TRACK

Demo de una app completa.

He usado un fork del vue-pokedex de @rubnvp (https://github.com/manufosela/vue-pokedex)

GRACIAS

@manufosela - manufosela@gmail.com