Find primary difference between Docker and Podman

Docker usa un daemon, un programa de fondo que corre imágenes y crea contenedores mientras Podman tiene una arquitectura daemonless y no necesita privilegios root para administrar contenedores. Los contenedores en Podman no tienen privilegios root por defecto, haciéndolos más seguros, aunque también pueden correrlos con privielgios root.

Docker es una herramienta todo en uno, con beneficios y desventajas que eso conlleva mientras Podman tiene un enfoque modular, dependiendo de herramientas externas para tareas específicas.

Create, build and run container with a Dockerfile

Dockerfile

```
Dockerfile > ...
      You, hace 5 minutos | 1 author (You)
1     FROM node:16
2
3     # Create app directory
4     WORKDIR /usr/src/app
5
6     # Install app dependencies
7     # A wildcard is used to ensure both package.json AND package-lock.json are copied
8     # where available (npm@5+)
9     COPY package*.json ./
10
11    RUN npm install
12    # If you are building your code for production
13    # RUN npm ci --only=production
14
15    # Bundle app source
16    COPY . .
17
18    EXPOSE 8080
19    CMD [ "node", "server.js" ]      You, hace 6 minutos • first commit
```

Docker image build

```
C:\Users\hp\Documents\bootcamp\devops\repos\dockerfile-test>docker build . -t dockerfile-test

[+] Building 18.9s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                                 1.3s
 => => transferring dockerfile: 439B                                                 0.0s
 => [internal] load .dockerignore                                                    1.5s
 => => transferring context: 67B                                                     0.0s
 => [internal] load metadata for docker.io/library/node:16                           2.3s
 => [internal] load build context                                                    0.4s
 => => transferring context: 41.91kB                                                 0.0s
 => [1/5] FROM docker.io/library/node:16@sha256:68fc9f749931453d5c8545521b021dd97267e0  0.0s
 => CACHED [2/5] WORKDIR /usr/src/app                                                0.0s
 => [3/5] COPY package*.json ./                                                      1.6s
 => [4/5] RUN npm install                                                            6.1s
 => [5/5] COPY . .                                                                   1.5s
 => exporting to image                                                               4.6s
 => => exporting layers                                                              3.8s
 => => writing image sha256:da4d5f9974f434def2f527bb43d5096bdbad36bb70774404cc7f761032  0.1s
 => => naming to docker.io/library/dockerfile-test                                   0.1s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Container running

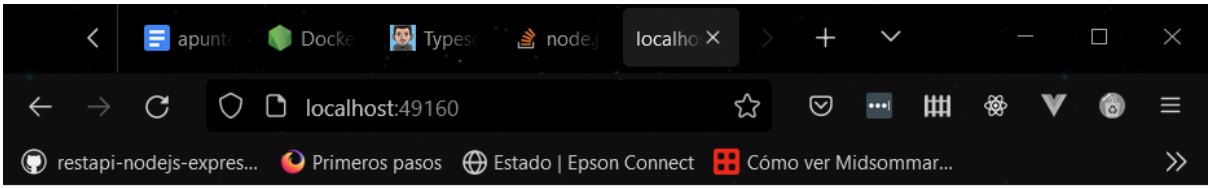
```
C:\Users\hp\Documents\bootcamp\devops\repos\dockerfile-test>docker run -p 49160:8080 -d dockerfile-t
est
8d99bb6853890b51979841f477ed78d9a508bd8f0eece66c7414e68aa8145aa1

C:\Users\hp\Documents\bootcamp\devops\repos\dockerfile-test>docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS          PORTS
                NAMES
8d99bb685389   dockerfile-test   "docker-entrypoint.s…"   34 seconds ago   Up 31 seconds   0.0.0.0:4
9160->8080/tcp   sharp_vaughan

C:\Users\hp\Documents\bootcamp\devops\repos\dockerfile-test>docker logs 8d9
Running on http://0.0.0.0:8080

C:\Users\hp\Documents\bootcamp\devops\repos\dockerfile-test>docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED              STATUS              PORTS                     NAMES
8d99bb685389   dockerfile-test   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:49160->8080/tcp   sharp_vaughan

C:\Users\hp\Documents\bootcamp\devops\repos\dockerfile-test>
```



Hello World