Prácticas Docker

1. Inspects

- Vamos a realizar alguna prueba el comando “inspect”
- Nos bajamos la imagen de NodeJS, uno de los productos más utilizados en Internet. NOTA: la salida puede ser un poco distinta, ya que depende las imágenes que os hayáis descargado previamente

```
docker pull node
Using default tag: latest
latest: Pulling from library/node
dbba69284b27: Already exists
9baf437a1bad: Already exists
6ade5c59e324: Already exists
b19a994f6d4c: Already exists
8fc2294f89de: Pull complete
0eec338961bd: Pull complete
6b0eb7b29093: Pull complete
9349bc5bacd1: Pull complete
135eddcd4b60: Pull complete
Digest:
sha256:0b553d28086d90b9b3be3339beb97401f8c0a83c17230a37ad9
9ff88fdad3b3f
Status: Downloaded newer image for node:latest
docker.io/library/node:latest
```

- Comprobamos que la tenemos

```
docker images | grep node
node              latest    20c0a0be5115   11 days ago    991MB
```

- Comprobamos las propiedades de la imagen. Debe salir bastante información

```
docker image inspect node
[
    {
```

```
        "Id":
"sha256:20c0a0be5115616fa6d27d6c72aefe8663340fd7f5fce9bc9d
62728b2efe7a75",
        "RepoTags": [
            "node:latest"
        ],
        "RepoDigests": [

"node@sha256:0b553d28086d90b9b3be3339beb97401f8c0a83c17230
a37ad99ff88fdad3b3f"
        ],
        "Parent": "",
        "Comment": "",
        "Created": "2022-03-31T01:09:38.988769786Z",
        "Container":
"68e036e4b449411cb75852f3df8517c4842d9df47a7d46ea9efa9df64
214caf2",
        "ContainerConfig": {
            "Hostname": "68e036e4b449",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,


….
….
…
```

- Podemos usar GREP para encontrar información más concreta

```
docker image inspect node | grep NODE_VERSION
                "NODE_VERSION=17.8.0",
                "NODE_VERSION=17.8.0",
```

- Lo mejor es mandar la salida a un fichero para inspeccionarlo después

```
docker image inspect node > node.json
```

- Creamos ahora un contenedor con esa imagen

```
docker run -d -it --name node1 node
9430087a9eeb39199eb292a95c8968a53f7008b3a4bfbf75f4e097d24988fc48

docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED
STATUS          PORTS     NAMES
9430087a9eeb   node      "docker-entrypoint.s…"   40 seconds ago   Up 38 seconds               node1
```

- Lanzamos un inspect contra el contenedor. También debe aparecer mucha información

```
docker inspect node1
[
    {
        "Id": "9430087a9eeb39199eb292a95c8968a53f7008b3a4bfbf75f4e097d24988fc48",
        "Created": "2022-04-11T02:59:46.78291111Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "node"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 39145,
```

```
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2022-04-11T02:59:47.603740769Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image": "sha256:20c0a0be5115616fa6d
….
….
```

- Podemos usar de nuevo el GREP. Por ejemplo, hay un campo que nos permite ver la dirección IP que ha asignado al contenedor

```
docker inspect node1 | grep IPAddress
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.4",
                    "IPAddress": "172.17.0.4",
```

- También podemos ver la memoria compartida asignada
-

```
docker inspect node1 | grep ShmSize
            "ShmSize": 67108864,
```

- Tenemos la posibilidad de utilizar un formato para encontrar determinada información, aunque eso nos obliga a conocer el nombre completo de la propiedad y su jerarquía. Se usa la opción –format. Se usan plantilla creada con el lenguaje GO
- Por ejemplo, para saber las direcciones IP que puede tener el contenedor:

```
docker inspect --format='{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' node1
172.17.0.4
```

- O donde deja el fichero log

```
docker inspect --format='{{.LogPath}}' node1
/var/lib/docker/containers/9430087a9eeb39199eb292a95c8968a
53f7008b3a4bfbf75f4e097d24988fc48/9430087a9eeb39199eb292a9
5c8968a53f7008b3a4bfbf75f4e097d24988fc48-json.log
```

- Y por último podemos ver la imagen en la que está basado

```
docker inspect --format='{{.Config.Image}}'  node1
node
```