Experiment 9 (Docker Metadata & Labels)

Name: Ankur
Roll No: R171218023

This command assigns a label called user with an ID to the container. This would allow us to query for all the containers running related to that particular user

```
$ docker run -l user=12345 -d redis
7c0eae59f9c7ce3e7e5e74e81849598dd5cf230fda2e8dff75cb90e58ab5b80e
```

1st command creates two labels in the file, one for the user and the second assigning a role and in 2nd command, the --label-file=<filename> option will create a label for each line in the file

```
$ echo 'user=123461' >> labels && echo 'role=cache' >> labels
$ docker run --label-file=labels -d redis
aa9e9138f631b1f1bc92363e3199f9d4b87eee41bd0a2baefc24bc32fa3c0306
```

Providing the running container's friendly name or hash id and querying all of it's metadata


```
$ docker inspect rd
[
    {
        "Id": "8c7ab0b2f7dc3b4b8c428956617280d8f917b093d68f5ebdb5f39571c1d1a3b4",
        "Created": "2021-05-02T16:04:27.550176441Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "redis-server"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 2464,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2021-05-02T16:04:27.984551845Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image": "sha256:4e8db158f18dc71307f95260e532df39a9b604b51d4e697468e82845c50cfe28",
        "ResolvConfPath": "/var/lib/docker/containers/8c7ab0b2f7dc3b4b8c428956617280d8f917b093d68f5ebdb5f39571c1d1a3b4/resolv
.conf",
        "HostnamePath": "/var/lib/docker/containers/8c7ab0b2f7dc3b4b8c428956617280d8f917b093d68f5ebdb5f39571c1d1a3b4/hostname
",
```

```
            "Networks": {
                "bridge": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": null,
                    "NetworkID": "ebb5e611abeb551587eb6db751b2077d289eba7e8c3715c6a49653a80994affa",
                    "EndpointID": "d1c43dfdb4a39f8da6540988c75d732f529e83dd839c15b451b06c0c23e7a753",
                    "Gateway": "172.18.0.1",
                    "IPAddress": "172.18.0.3",
                    "IPPrefixLen": 24,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "02:42:ac:12:00:03",
                    "DriverOpts": null
                }
            }
        }
    }
]
```


Using the -f option filtering the JSON response to just the labels section we're interested in

```
$ docker inspect -f "{{json .Config.Labels }}" rd
{"com.katacoda.created":"automatically","com.katacoda.private-msg":"magic","user":"scrapbook"}
```

Inspecting images works in the same way however the JSON format is slightly different, naming it ContainerConfig instead of Config

```
$ docker inspect -f "{{json .ContainerConfig.Labels }}" katacoda-label-example
{"com.katacoda.build-date":"2015-07-01T10:47:29Z","com.katacoda.course":"Docker","com.katacoda.private-msg":"HelloWorld","com
.katacoda.version":"0.0.5","vendor":"Katacoda"}
```

The query below will return all the containers which have a user label key with the value katacoda.

```
$ docker ps --filter "label=user=scrapbook"
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
8c7ab0b2f7dc        redis               "docker-entrypoint.s…"   45 seconds ago      Up 44 seconds       6379/tcp
rd
```

The same filter approach can be applied to images based on the labels used when the image was built.

```
$ docker images --filter "label=vendor=Katacoda"
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
katacoda-label-example  latest              5cd97d6a1a7d        43 seconds ago      84.1MB
$
```