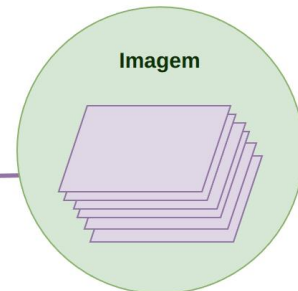




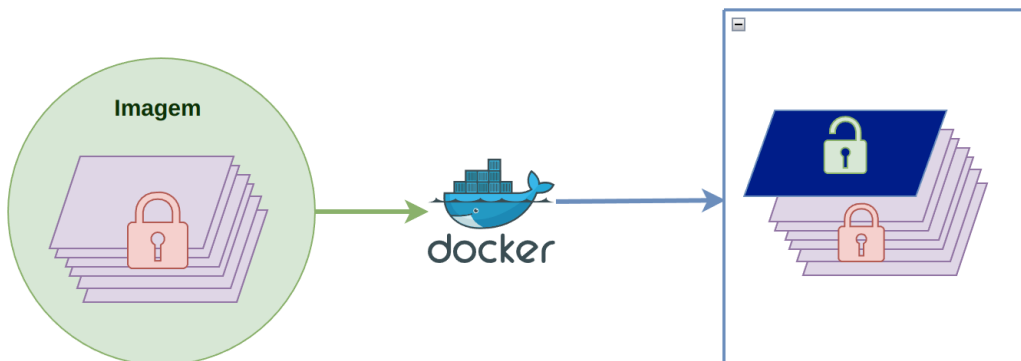
O que são Imagens

Uma **imagem** é composta por camadas, quando as camadas são empilhadas elas formam regras de execução do **container**. Falando assim é meio abstrato, então vou usar o **elemento visual**, **camadas** são os “comandos” que **compõe a imagem**, então se você usar o comando **“docker history [idImagem]”** as camadas dela podem ser visualizadas:

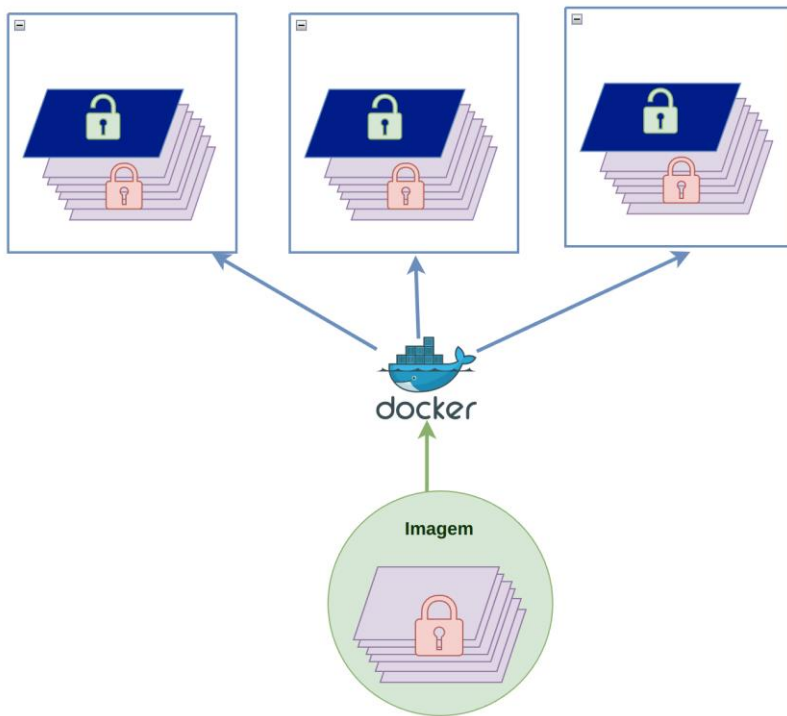
```
lucas-ubuntu@lucasubuntu-desktop: ~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest             27941809078c       11 days ago        77.8MB
hello-world         latest             feb5d9fea6a5       8 months ago       13.3kB
dockersamples/static-site latest             f589ccde7957       6 years ago        191MB
lucas-ubuntu@lucasubuntu-desktop: ~$ docker history f589ccde7957
IMAGE      CREATED BY                                     SIZE      COMMENT
f589ccde7957 6 years ago /bin/sh -c #(nop) CMD ["/bin/sh" "-c" "cd /u_ 0B
missing> 6 years ago /bin/sh -c #(nop) WORKDIR /usr/share/nginx/h_ 0B
missing> 6 years ago /bin/sh -c #(nop) COPY file:c8203f6bre2ff6ba_ 8.75kB
missing> 6 years ago /bin/sh -c mkdir -p /usr/share/nginx/html 0B
missing> 6 years ago /bin/sh -c #(nop) ENV AUTHOR=Docker 0B
missing> 6 years ago /bin/sh -c #(nop) CMD ["nginx" "-g" "daemon _ 0B
missing> 6 years ago /bin/sh -c #(nop) EXPOSE 443/tcp 80/tcp 0B
missing> 6 years ago /bin/sh -c ln -sf /dev/stdout /var/log/nginx. 22B
missing> 6 years ago /bin/sh -c apt-key adv --keyserver hkp://pdp. 65.4MB
missing> 6 years ago /bin/sh -c #(nop) ENV NGINX_VERSION=1.9.12-1_ 0B
missing> 6 years ago /bin/sh -c #(nop) MAINTAINER NGINX Docker Ma_ 0B
missing> 6 years ago /bin/sh -c #(nop) CMD ["/bin/bash"] 0B
missing> 6 years ago /bin/sh -c #(nop) ADD file:b5391cb13172fb513_ 125MB
lucas-ubuntu@lucasubuntu-desktop: ~$
```



As imagens são **IMUTÁVEIS**, o que significa que são apenas para leitura, uma vez **montada não podem ser alteradas**. Mas como é possível escrever dentro de um **container** se as **imagens** são read only? Isso porque o **container** é uma **camada extra** que **permite a leitura/escrita**:



Esse conceito de camadas também ajuda a entender porque containers são mais leves que máquinas virtuais. Basicamente falando, o **Docker** é inteligente o bastante pra reaproveitar **camadas** já existentes na máquina, então toda vez que um **container** é montado, as **camadas** em comum de uma **imagem** são reaproveitadas:



Como são criadas Imagens

O Docker tem um arquivo especial que define como uma imagem deve ser criada, esse arquivo é o **Dockerfile**. O interessante é que **imagens** podem ser criadas com base em imagens já existentes, onde você pode definir **variáveis de ambiente, portas para expor o container, diretórios principais, comandos e etc**:

```
Dockerfile X
Dockerfile
1 #define a imagem base da imagem que vai ser criada
2 FROM node:14
3
4 #cria uma pasta principal de trabalho dentro do container
5 WORKDIR /app-node
6
7 #variavel de ambiente durante a construção da imagem/ build do container
8 ARG PORT_BUILD=6000
9
10 #varial de ambiente dentro do container
11 ENV PORT=$PORT_BUILD
12
13 # deixa claro o valor da porta durante a listagem do container
14 EXPOSE $PORT_BUILD
15
16 # copia o que ta no diretório raiz do dockerfile pra dentro do workdir definido
17 COPY . .
18
19 # aqui eu to executando um comando de node na pasta de workdir, então ele sabe se virar com esse comando
20 RUN npm install
21
22 # executa toda vez que o container é iniciado
23 ENTRYPOINT npm start
```

Esse arquivo pode ser transformado em imagem com o comando ***“docker build -t xxxxxxxxxxxx”*** o **-t** é uma flag para podermos nomear a imagem.