

```
STATUS  
ACCOUNT  
STATUS  
ng"==>f.type(a),F=function(a){return p(a)&&0<a.indexOf("%")},l=function(a,d){var  
nt(a,10)||0;d&&F(a)&&(e=b.setViewport()[d]/100);return Math.ceil(e)},x=function(a,b){return  
ox";f.extend(b,{version:"2.1.4",defaults:{padding:15,margin:20,width:800,  
00,minWidth:100,minHeight:100,maxWidth:9999,maxHeight:9999,autoSize:!0,autoHeight:!1,autoWidth:!1,autoResize  
center:!s,fitToView:!0,aspectRatio:!1,topRatio:0.5,leftRatio:0.5,scrolling:"auto",wrapCSS:"",arrows:!0,close  
oseClick:!1,nextClick:!1,mouseWheel:!0,autoPlay:!1,playSpeed:3E3,preload:3,modal:!1,loop:!0,ajax:{dataType:  
aders:{'X-fancyBox':!0}},iframe:{scrolling:"auto",preload:!0},swf:{wmode:"transparent",allowfullscreen:"true  
criptaccess:"always"},keys:{next:{13:"left",  
0:"left",40:"up"},prev:{8:"right",37:"down",38:"up"},close:[27],play:[32],toggle:[70]},directi  
"left",prev:"right"},scrollOutside:!0,index:0,type:null,href:null,content:null,title:null,tpl:{wrap:'<div  
cybox-wrap" tabIndex="-1"><div class="fancybox-skin"><div class="fancybox-outer"><div  
cybox-inner"></div></div></div>',image:' Dockerfile > ...  
1 FROM node:18  
2  
3 WORKDIR /app  
4  
5 COPY package.json ./  
6  
7 RUN npm install  
8  
9 COPY . .  
10  
11 EXPOSE 80  
12  
13 CMD [ "node", "app.js" ]  
14
```

Figura 1 – Dockerfile - Layers
Fonte: Elaborado pelo autor (2023)

O arquivo mostrado tem sete (7) camadas (layers). Ao criar uma imagem, as próximas alterações no arquivo Dockerfile serão usadas como cache (dependendo em qual camada ocorreu a alteração).

Por exemplo, se adicionamos um novo arquivo na raiz do projeto, a camada “COPY”, da linha 9, sofrerá alterações. As camadas anteriores a ela, serão executadas como cache, o que agiliza muito a alteração da imagem. Após a alteração dessa camada, todas as outras camadas serão criadas novamente do zero. Um exemplo de saída com o cenário descrito:

```
Sending build context to Docker daemon 2.606MB
Step 1/7 : FROM node:18
18: Pulling from library/node
90e5e7d8b87a: Pull complete
27e1a8ca91d3: Pull complete
d3a767d1d12e: Pull complete
711be5dc5044: Pull complete
22956530cc64: Pull complete
d38ebdae17cd: Pull complete
bfaacda23df76: Pull complete
4b9302a8baa0: Pull complete
Digest: sha256:a17842484dd30af97540e5416c9a62943c709583977ba41481d601ecffb7f31b
Status: Downloaded newer image for node:18
--> 219cd50149ac
Step 2/7 : WORKDIR /app
--> Running in 39b0136c06c7
Removing intermediate container 39b0136c06c7
--> 3328d0b69116
Step 3/7 : COPY package.json .
--> 04a5cfaf7df26
Step 4/7 : RUN npm install --silent
--> Running in e1d94b96dcf3
npm notice
npm notice New major version of npm available! 9.8.1 -> 10.2.4
npm notice Changelog: <https://github.com/npm/cli/releases/tag/v10.2.4>
npm notice Run `npm install -g npm@10.2.4` to update!
npm notice
Removing intermediate container e1d94b96dcf3
--> 3e0b864763ad
Step 5/7 : COPY . .
--> 605dba4b8b8a
Step 6/7 : EXPOSE 80
--> Running in ceb4dda45c33
Removing intermediate container ceb4dda45c33
--> f1f28849f3d6
Step 7/7 : CMD [ "node", "app.js" ]
--> Running in f3c858a858a9
Removing intermediate container f3c858a858a9
--> ede35c4947ce
Successfully built ede35c4947ce
```

Figura 2 – Dockerfile – Layers Build
Fonte: Elaborado pelo autor (2023)

A imagem foi criada e, como é a primeira versão, demoramos cerca de 55 segundos para a criação. Já a figura “Dockerfile – Layers Build Cached” demorou cerca de 15 segundos, devido ao uso do cache nas camadas que não sofreram alterações:

```
Sending build context to Docker daemon 2.606MB
Step 1/7 : FROM node:18
--> 219cd50149ac
Step 2/7 : WORKDIR /app
--> Using cache
--> 309366833734
Step 3/7 : COPY package.json ./
--> Using cache
--> a9e20c315c90
Step 4/7 : RUN npm install --silent
--> Using cache
--> 4d40d5172857
Step 5/7 : COPY .
--> 28b753dbe8fd
Step 6/7 : EXPOSE 80
--> Running in 65afda0fed90
Removing intermediate container 65afda0fed90
--> 15ff51405053
Step 7/7 : CMD [ "node", "app.js" ]
--> Running in 28c73daa7f58
Removing intermediate container 28c73daa7f58
--> 2b47a65465a7
Successfully built 2b47a65465a7
```

Figura 3 – Dockerfile – Layers Build Cached

Fonte: Elaborado pelo autor (2023)

Dessa maneira, as camadas devem ser pensadas para obter essa estrutura, facilitando e agilizando os testes e o desenvolvimento locais. O uso do cache também se aplica aos repositórios on-line, como Docker Hub, AWS ECR, Azure Container Registry e Google Cloud Container Registry.

Outra boa prática já citada é o uso de ARGs (Argumentos) para não expor valores sensíveis em um arquivo Dockerfile. Os argumentos são passados no momento do build da imagem, que recebe esses argumentos e os insere como valor do argumento definido no arquivo Dockerfile, sendo possível utilizá-lo junto da instrução ENV. Veja um exemplo demonstrando essa boa prática:

```
1  FROM node:16.20.2-buster-slim
2
3  WORKDIR /app
4
5  ARG ARG_NODE_ENV
6
7  ENV NODE_ENV=${ARG_NODE_ENV}
8
9  COPY package.json ./
10
11 RUN npm install
12
13 COPY . .
14
15 EXPOSE 80
16
17 CMD [ "node", "app.js" ]
```

Figura 4 – Dockerfile – ARG com ENV
Fonte: Elaborado pelo autor (2023)

```
→ node-app docker build --build-arg ARG_NODE_ENV=Development .
failed to fetch metadata: fork/exec /usr/local/lib/docker/cli-plugins/docker-buildx: no such file or
directory

DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 2.606MB
Step 1/7 : FROM node:18
--> 219cd50149ac
Step 2/7 : WORKDIR /app
--> Using cache
--> 3328d0b69116
Step 3/7 : COPY package.json ./
--> Using cache
--> 04a5cfa7df26
Step 4/7 : RUN npm install --silent
--> Using cache
--> 3e0b864763ad
Step 5/7 : COPY . .
--> Using cache
--> 605dba4b8b8a
Step 6/7 : EXPOSE 80
--> Using cache
--> f1f28849f3d6
Step 7/7 : CMD [ "node", "app.js" ]
--> Using cache
--> ede35c4947ce
[Warning] One or more build-args [ARG_NODE_ENV] were not consumed
Successfully built ede35c4947ce
→ node-app
```

Figura 5 – Dockerfile – build-arg
Fonte: Elaborado pelo autor (2023)

Agora, por exemplo, podemos ver e entender sobre layers de imagens que estão no Docker Hub. A figura “Docker Hub – Layers” mostra os layers de uma imagem node:

The screenshot shows the Docker Hub interface for the image `node:its-alpine3.18`. At the top, there's a summary card with the image name, a green 'node' logo, and a 'Scout' badge. Below it, details like OS/ARCH (linux/amd64), COMPRESSED SIZE (45.5 MB), LAST PUSHED (11 hours ago by [dolankey](#)), TYPE (Image), and VULNERABILITIES (No Vulnerabilities Found). A purple banner at the top says 'Get advanced image analysis for your own images with Docker Scout. [Learn more](#) and [upgrade](#)'. The main content area has tabs for 'Images (1)', 'Vulnerabilities (0)', and 'Packages (19)'. The 'Packages (19)' tab is selected. It shows a list of packages from Alpine Linux, including alpine/busybox, alpine/apk-tools, and alpine/ca-certificates. On the left, a red box highlights the 'Image hierarchy' section, which displays the layers of the image. The layers are listed as follows:

	Layer ID	Action	File	Size	Status
FROM	0	ADD file:	fc714080c3bcbce7ac746a10d7b4355ffa3...	3.4 MB	✓
ALL	1	CMD	['/bin/sh']	0 B	✓
	2	ENV	NODE_VERSION=20.10.0	0 B	✓
	3	/bin/sh -c	addgroup -g 1000 node && adduser -u 100...	41.96 MB	✓
	4	ENV	YARN_VERSION=1.22.19	0 B	✓
	5	/bin/sh -c	apk add --no-cache --virtual .build-deps-ya...	2.34 MB	✓
	6	COPY	file:4d192565a7220e135cab6c77fbc1c73211...	448 B	✓
	7	ENTRYPOINT	["docker-entrypoint.sh"]	0 B	✓
	8	CMD	["node"]	0 B	✓

Figura 6 – Docker Hub – Layers
Fonte: hub.docker.com (2023), adaptado pelo autor (2023)

O QUE VOCÊ VIU NESTA AULA?

Nesta aula, você viu o que são e como funcionam os layers de uma imagem docker, além da combinação das instruções ARG e ENV do Dockerfile. Esperamos que vocês tenham gostado desta aula! Em caso de dúvidas, nos procurem no Discord e não deixem de assistir às videoaulas. Até mais!



REFERÊNCIAS

BOAS práticas para criar imagens Docker eficientes, seguras e escaláveis. **Gaspar Barancelli.** 2023. Disponível em: <<https://www.gasparbarancelli.com/post/boas-praticas-para-criar-imagens-docker-eficientes-seguras-e-escalaveis>>. Acesso em: 09 jan. 2024.

DOCKER Hub – Node Tags and Layers. **HUB.DOCKER.COM.** [s.d]. Disponível em: <https://hub.docker.com/_/node/tags>. Acesso em: 09 jan. 2024.



PALAVRAS-CHAVE

ARG. Camadas. Dockerfile. Layers.



```
STATUS  
ACCOUNT  
STATUS  
  
class Banner  
attr_accessible :horiz, :link, :visible, :image, :position  
has_attached_file :image, styles: { vert: '220'  
before_create :assign_position  
  
ing==="f.type(a)},F=function(a){return p(a)&&0<a.indexOf("%")},l=function(a,d){var  
int(a,10)||0;d&&F(a)&&(e=b.getViewport()[d]/100);return Math.ceil(e)},x=function(a,b){return  
px"};f.extend(b,{version:"2.1.4",defaults:{padding:15,margin:20,width:800,  
00,minWidth:100,minHeight:100,maxWidth:9999,maxHeight:9999,autoSize:!0,autoHeight:!1,autoWidth:!1,autoResize:  
Center:!s,fitToView:!0,aspectRatio:!1,topRatio:0.5,leftRatio:0.5,scrolling:"auto",wrapCSS:"",arrows:!0,close  
closeClick:!1,nextClick:!1,mouseWheel:!0,autoPlay:!1,playSpeed:3E3,preload:3,modal:!1,loop:!0,ajax:{dataType:  
adlers:{'X-fancyBox':!0},iframe:{scrolling:"auto",preload:!0},swf:{wmode:"transparent",allowfullscreen:"true",  
scriptaccess:"always"},keys:{next:{13:"left",  
9:"left",40:"up"},prev:{8:"right",33:"down",37:"right",38:"down"},close:[27],play:[32],toggle:[70]},directi  
"left",prev:"right"},scrollOutside:!0,index:0,type:null,href:null,content:null,title:null,tpl:{wrap:'<div  
cybox-wrap" tabIndex="-1"><div class="fancybox-skin"><div class="fancybox-outer"><div  
cybox-inner"></div></div></div>',image:'  
  
POS TECH  
  
class Banner < ActiveRecord::Base  
attr_accessible :horiz, :link, :visible, :image, :position  
has_attached_file :image, styles: { vert: '220'  
before_create :assign_position  
  
protected  
  
def assign_position  
max = Banner.maximum(:position)  
self.position = max ? max + 1 : 0
```