

Hospedar projetos de projetos desenvolvidos em HTML + CSS é uma alternativa para a soluções cujo único sentido é fazer a divulgação de uma empresa ou serviço, disponibilizar na internet formas de contato, ou seja, ser um canal estático que não tem a necessidade de operações e processamentos robustos.

Existem alguns servidores como Apache e Nginx que permitem a publicação de projetos inteiros de sites, que até podem prover responsividade (ou seja, se adaptam caso a exibição seja em um dispositivo portátil).

O Apache é um servidor web livre que permite que pessoas ou empresas criem uma estrutura que provê serviços para a hospedagem de projetos baseados em HTML + CSS.

O exercício consiste em:

- 1) Criar um pequeno projeto baseado HTML, que será padrão.
- 2) Criar um dockerfile que permita criar a imagem Docker para prover o serviço web
- 3) Criar um contêiner que deixe o serviço no ar (local), permitindo modificações no projeto

## **Parte 1**

Como, cada vez menos, se utilizam *frames* para a construção de um HTML, pesquisar a sintaxe e construir os seguintes arquivos (Criação pode ser no bloco de notas). Para pesquisar Tags HTML, usar <https://www.w3schools.com/html/>

O cabeçalho deve trazer o título: Projeto HTML Docker (em todas as páginas)

O menu, que deve aparecer em todas as páginas (Não precisa ser include, pode ser hardcoded) deve ter links que permitam acessar as páginas index.html, docker.html, comandos.html

index.html

---

[Index](#) [Docker](#) [Principais Comandos Docker](#)

Esta página apresentará uma breve apresentação de **Docker** e seus comandos

docker.html

---

[Index](#) [Docker](#) [Principais Comandos Docker](#)

A tecnologia Docker usa o kernel do Linux e recursos do kernel como Cgroups e namespaces para segregar processos. Assim, eles podem ser executados de maneira independente. O objetivo dos containers é criar essa independência: a habilidade de executar diversos processos e aplicações separadamente para utilizar melhor a infraestrutura e, ao mesmo tempo, manter a segurança que você teria em sistemas separados.

As ferramentas de container, incluindo o Docker, fornecem um modelo de implantação com base em imagem. Isso facilita o compartilhamento de uma aplicação ou conjunto de serviços, incluindo todas as dependências deles em vários ambientes. O Docker também automatiza a implantação da aplicação (ou de conjuntos de processos que constituem uma aplicação) dentro desse ambiente de container.

comandos.html

[Index](#) [Docker](#) [Principais Comandos Docker](#)

docker build – A partir de instruções de um arquivo Dockerfile eu possa criar uma imagem.  
docker exec – Executa uma instrução dentro do container que está rodando sem precisar atachar nele.  
docker images – Lista as imagens disponíveis no host.  
docker logs – Exibe os logs de um container.  
docker ps – Lista todos os containers.  
docker pull – Faz o pull de uma imagem a partir de um servidor de registry.  
docker push – Faz o push de uma imagem a partir de um servidor de registry.  
docker restart – Restarta um container que está rodando ou parado.  
docker rm – Remove um ou mais containers.  
docker rmi – Remove uma ou mais imagens.  
docker run – Executa um comando em um novo container.  
docker start – Inicia um container que esteja parado.  
docker stop – Para um container que esteja rodando.

## **Parte 2**

Criar um Dockerfile:

- a. A partir da imagem httpd;
- b. Deixando seu nome como label;
- c. Rodando apt-get update;
- d. Criando (mkdir -p) a pasta /usr/local/apache2/htdocs/htmldocker/
- e. Criando (mkdir -p) a pasta /var/htmldocker/
- f. Dando permissão total (chmod -R 777 path) para a pasta /usr/local/apache2/htdocs/htmldocker/
- g. Dando permissão total (chmod -R 777 path) para a pasta /var/htmldocker/
- h. Copiando os arquivos da pasta para /usr/local/apache2/htdocs/htmldocker;
- i. Apagando o arquivo Dockerfile da pasta /usr/local/apache2/htdocs/htmldocker ;
- j. Deixando como diretório de trabalho a pasta /usr/local/apache2/htdocs/htmldocker;
- k. Expondo a porta 80;
- l. Deixando como volume mapeado a pasta /var/htmldocker/;

A imagem deve chamar htmldockering

Ao criar a pasta htmldocker como subpasta de htdocs, a URL do projeto HTML deverá ser http://caminho/htmldocker/

## **Parte 3**

Ativar o sudo: sudo su (e colocar a senha)

Criar uma pasta /usr/local/htdocs

Dar permissão total a ela: CHMOD -R 777 /usr/local/htdocs

Sair do sudo: exit

Criar um contêiner:

- a. Em background
- b. Com nome de htmldocker
- c. Mapeando a porta 32000 local na porta 80 do contêiner
- d. Mapear o volume /usr/local/htdocs local na pasta /var/htmldocker/ do contêiner
- e. Baseado na imagem htmldockering

O Docker não trabalha adequadamente se há o mapeamento de um volume em uma pasta já com arquivos, por isso mapeamos /var/htmldocker como pasta interna e não a ../htdocs/htmldocker

Mapeando a porta 32000 local na porta 80 do contêiner, nossa URL deve ficar:

http://localhost:32000/htmldocker/

Podendo rodar esse link no navegador. Caso não apareça o conteúdo de index.html, rever a criação da imagem e do contêiner.

#### Parte 4

Criar na pasta local /usr/local/htdocs, um arquivo dockerfile.html, como abaixo

[Index](#) [Docker](#) [Principais Comandos Docker](#) [Principais Comandos Dockerfile](#)

## DOCKERFILE

Arquivo: **Dockerfile**

Para gerar uma imagem a partir de Dockerfile: `docker build -t nome_imagem .`

*Comandos Dockerfile:*

**FROM** – Chamada da imagem que será base para a construção da minha imagem

**RUN** – Permitir a execução de comandos na montagem da imagem

**CMD x ENTRYPOINT** – Executam um determinado comando na inicialização do contêiner gerado a partir da imagem customizada por Dockerfile

**COPY x ADD** – Permitem a cópia de 1 ou mais arquivos para uma pasta específica da imagem

**ADD** – Serve pra copiar arquivos compactados (tar), descompactando na cópia

**EXPOSE** – Informa qual porta deverá ser liberada

**VOLUME** – Determinar o ponto de montagem para uma pasta que ficará disponível entre host e contêiner

**WORKDIR** – Determina o diretório de trabalho. Na criação do contêiner a partir da imagem customizada, passa a ser o diretório padrão

**USER** – Define o usuário que executa comandos

**LABEL** – Define o mantenedor da imagem

Mudar, sem entrar no contêiner, os menus das outras páginas html para terem o link do dockerfile.html, como abaixo

---

[Index](#) [Docker](#) [Principais Comandos Docker](#) [Principais Comandos Dockerfile](#)

Esta página apresentará uma breve apresentação de **Docker** e seus comandos

---

[Index](#) [Docker](#) [Principais Comandos Docker](#) [Principais Comandos Dockerfile](#)

A tecnologia Docker usa o kernel do Linux e recursos do kernel como Cgroups e namespaces para segregar processos. Assim, eles podem ser executados de maneira independente. O objetivo dos containers é criar essa independência: a habilidade de executar diversos processos e aplicações separadamente para utilizar melhor a infraestrutura e, ao mesmo tempo, manter a segurança que você teria em sistemas separados.

As ferramentas de container, incluindo o Docker, fornecem um modelo de implantação com base em imagem. Isso facilita o compartilhamento de uma aplicação ou conjunto de serviços, incluindo todas as dependências deles em vários ambientes. O Docker também automatiza a implantação da aplicação (ou de conjuntos de processos que constituem uma aplicação) dentro desse ambiente de container.

[Index](#) [Docker](#) [Principais Comandos Docker](#) [Principais Comandos Dockerfile](#)

docker build – A partir de instruções de um arquivo Dockerfile eu possa criar uma imagem.  
docker exec – Executa uma instrução dentro do container que está rodando sem precisar atachar nele.  
docker images – Lista as imagens disponíveis no host.  
docker logs – Exibe os logs de um container.  
docker ps – Lista todos os containers.  
docker pull – Faz o pull de uma imagem a partir de um servidor de registry.  
docker push – Faz o push de uma imagem a partir de um servidor de registry.  
docker restart – Restarta um container que está rodando ou parado.  
docker rm – Remove um ou mais containeres.  
docker rmi – Remove uma ou mais imagens.  
docker run – Executa um comando em um novo container.  
docker start – Inicia um container que esteja parado.  
docker stop – Para um container que esteja rodando.

Copiar os arquivos html (todos) para a pasta local /usr/local/htdocs

Os arquivos estarão no contêiner, mas ainda não estarão disponíveis no servidor apache

Rodar o comando Docker exec para copiar do volume mapeado do contêiner para a pasta do projeto web.

Docker exec -it nome\_do\_container comando\_a\_odar\_dentro\_do\_container

```
docker exec -it htmldocker cp -r -v /var/htmldocker/. /usr/local/apache2/htdocs/htmldocker
```

O comando `cp -r -v /var/htmldocker/. /usr/local/apache2/htdocs/htmldocker` significa:

`cp` – copiar

`-r` – Recursivamente todos os arquivos da pasta

`-v` – Verbose (Mostrando o que foi feito)

`/var/htmldocker/.` – Caminho dos arquivos dentro do contêiner (Não esquecer o ponto no final)

`/usr/local/apache2/htdocs/htmldocker` – Pasta de destino dos arquivos

- O Linux, por padrão sobrescreve arquivos que tiverem o mesmo nome.

Rodar novamente no navegador (Não há necessidade de reiniciar o apache quando atualiza arquivos HTML)

Entregar:

- Dockerfile
- Comando docker run de criação do contêiner
- Print das páginas logo na sequência da criação do contêiner
- Print das páginas logo na sequência da modificação dos arquivos e atualização no Apache Web Server;