

Tutorial: implantação de Servidor Web NGINX em container Docker no VirtualBox com Linux

Daiane Maria dos Santos Ribeiro¹, Nereu Vítor Pereira Lima²

¹Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano –
IFSertãoPE Campus Salgueiro

BR 232, km 508, sentido Recife – 56.000-000 – Salgueiro – PE – Brasil

{daiane.ribeiro@ifsertao-pe.edu.br, nereu.vitor@aluno.ifsertao-pe.edu.br}

1. Introdução

Um servidor web é um sistema de software e hardware responsável por receber, interpretar e responder a requisições feitas por clientes via protocolo de transferência de hipertexto (HTTP, Hypertext Transfer Protocol) ou protocolo de transferência de hipertexto seguro (HTTPS, Hypertext Transfer Protocol Secure), geralmente entregando páginas web e serviços digitais (W3C, 2023). Existem inúmeros servidores web disponíveis para uso, mas o NGINX é um dos servidores mais adequado para ser instalado em um computador de uso pessoal, pois ele apresenta algumas vantagens em relação a outros servidores web, como por exemplo, excelente aproveitamento de CPU e RAM, ótimo para lidar com múltiplas conexões simultâneas, muito seguro e eficiente, ideal para ambientes de produção com WordPress, Laravel, APIs REST etc, alto desempenho mesmo em máquinas mais modestas, entre outras.

Para ilustrar como um servidor web funciona, nós iremos apresentar nesse trabalho um tutorial para criação e utilização de um container com o NGINX, que será utilizado a partir do Docker na máquina VirtualBox com o sistema operacional Linux. Ademais, apresentaremos, também, como disponibilizar a atividade desenvolvida no GitHub e no DockerHub.

O link de acesso do projeto desenvolvido nesse trabalho no GitHub é: <https://github.com/daianesr35/servidor-nginx-docker> e no DockerHub é: <https://hub.docker.com/repositories/daianesr35>.

Antes de apresentar o tutorial, faremos nas subseções a seguir, uma breve explicação sobre as tecnologias usadas nessa atividade.

1.1. Container com NGINX

Um container é uma unidade executável de software que empacota o código da aplicação juntamente com as suas bibliotecas e dependências para o seu funcionamento. Dessa forma, ele permite que o código seja executado em qualquer ambiente de computação, seja ele Desktop, TI tradicional ou em infraestrutura de nuvem (SUSNJARA, SMALLEY, 2025).

Os contêineres funcionam por meio da virtualização do Sistema Operacional (SO), no qual os recursos do kernel do SO podem ser utilizados para isolar processos e controlar a quantidade de memória, CPU e disco que esses processos podem utilizar. Oposto ao que as máquinas virtuais (MVs) realizam, a virtualização do sistema operacional (normalmente o Linux) realizada pelos contêineres, garante que somente a aplicação, suas bibliotecas, arquivos de configuração e dependências estejam presentes no interior do mesmo. Por conta da ausência do sistema operacional hospedeiro, tecnologia utilizada pelas máquinas virtuais, os contêineres tendem a serem mais leves, e, portanto, mais rápidas em comparação às MVs (SUSNJARA, SMALLEY, 2025). Ademais, para tornar o processo de containerização mais rápido e fácil é importante usar o Docker.

1.2. Docker

O Docker é uma plataforma de código aberto que permite aos desenvolvedores construir, implementar, executar, realizar atualizações e o gerenciar contêineres (SUSNJARA, SMALLEY, 2025).

Utilizar o Docker é muito importante, pois além dele tornar o processo de containerização mais rápido e fácil, também desempenha um papel crucial no desenvolvimento moderno de software, mais especificamente na área de microsserviços (SUSNJARA, SMALLEY, 2025).

A utilização do Docker para este trabalho foi realizada para facilitar e tornar ágil a criação, implementação e a gestão de um container com NGINX e as suas dependências, para que o mesmo atue como um servidor web, independente do ambiente que esteja sendo utilizado. Além disso, decidimos usar o GitHub e o DockerHub para disponibilizar o container com NGINX criado nesse trabalho.

1.3. GitHub DockerHub

O Github é uma plataforma baseada em nuvem onde é possível, armazenar, compartilhar e trabalhar com outras pessoas para escrever códigos. O uso do GitHub se dá por meio de repositórios remotos que podem ser públicos ou privados e podem receber solicitações de mudanças no projeto através de pull requests, onde o proprietário decide se aceitará ou não. Recebe, também, dúvidas sobre o projeto que podem ser esclarecidas através de issues criadas pelos os usuários (GITHUB, 2025).

De maneira análoga, o DockerHub é um repositório online onde é possível buscar, armazenar e compartilhar imagens Docker, facilitando o uso de containers em qualquer lugar.

Dessa maneira, escolhemos essas duas plataformas para disponibilizar o container com NGINX criado nesse trabalho. Além do professor da disciplina ter a possibilidade de acompanhar o desenvolvimento da atividade.

2. Passo a passo da criação e uso do container com NGINX no VirtualBox

Nesta seção iremos apresentar as etapas que realizamos desde a inicialização da Máquina Virtual (MV), do uso do servidor NGINX dentro do container no VirtualBox, até a publicação do container no GitHub.

2.1. Etapa 1 - Instalando o Docker na MV com Linux

1. O VirtualBox e Ubuntu Server já estava instalado no notebook. Então, nós inicializamos a MV o logamos no Ubuntu.

2. Em seguida, abrimos o terminal e executamos os comandos:

- `sudo apt update` – Esse comando atualizou a lista de pacotes disponíveis nos repositórios do Ubuntu. Isso é importante, porque garante que o sistema saiba quais são as versões mais recentes dos programas antes de instalar qualquer coisa, como mostra a figura 1.

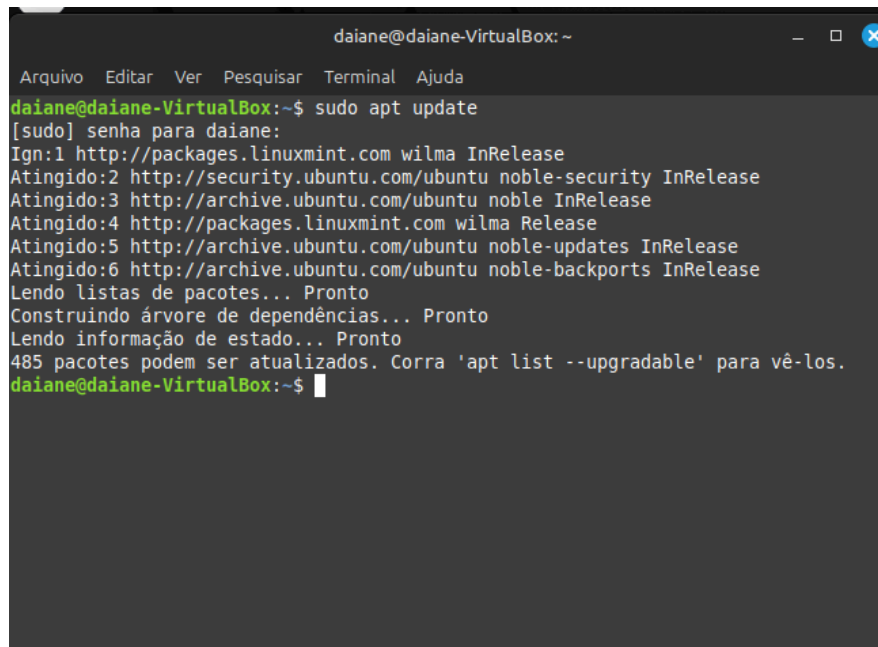
A screenshot of a terminal window titled 'daiane@daiane-VirtualBox: ~'. The terminal shows the command 'sudo apt update' being executed. The output includes prompts for a password, followed by several lines of repository information and status updates, such as 'Ign:1 http://packages.linuxmint.com wilma InRelease', 'Atingido:2 http://security.ubuntu.com/ubuntu noble-security InRelease', and 'Lendo listas de pacotes... Pronto'. The final line indicates that 485 packages can be updated and suggests running 'apt list --upgradable' to see them. The prompt returns to 'daiane@daiane-VirtualBox:~\$'.

Figura 1. Atualização de listas de pacotes no Ubuntu.

- `sudo apt install -y docker.io` – Esse commando instalou o pacote `docker.io`, que é o Docker disponível nos repositórios do Ubuntu. Assim, o Docker foi instalado na MV, como ilustra a figura 2, permitindo a criação e gerenciamento de containers.

```
daiane@daiane-VirtualBox: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
Lendo listas de pacotes... Pronto  
Construindo árvore de dependências... Pronto  
Lendo informação de estado... Pronto  
485 pacotes podem ser atualizados. Corra 'apt list --upgradable' para vê-los.  
daiane@daiane-VirtualBox:~$ sudo apt install -y docker.io  
Lendo listas de pacotes... Pronto  
Construindo árvore de dependências... Pronto  
Lendo informação de estado... Pronto  
Os pacotes adicionais seguintes serão instalados:  
  bridge-utils containerd git git-man liberror-perl pigz runc ubuntu-fan  
Pacotes sugeridos:  
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx  
  docker-compose v2 docker-doc rinse git-daemon-run | git-daemon-sysvinit  
  git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn  
Os NOVOS pacotes a seguir serão instalados:  
  bridge-utils containerd docker.io git git-man liberror-perl pigz runc  
  ubuntu-fan  
0 pacotes atualizados, 9 pacotes novos instalados, 0 a serem removidos e 485 não atualizados.  
E preciso baixar 83,7 MB de arquivos.  
Depois desta operação, 323 MB adicionais de espaço em disco serão usados.  
Obter:1 http://archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65,6 kB]  
Obter:2 http://archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33,9 kB]  
Obter:3 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.2.5-0ubuntu1~24.04.1 [8.043 kB]  
Obter:4 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.27-0ubuntu1~24.04.1 [37,7 MB]  
Obter:5 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 docker.io amd64 27.5.1-0ubuntu3~24.04.2 [33,0 MB]  
Ign:5 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 docker.io amd64 27.5.1-0ubuntu3~24.04.2  
Obter:6 http://archive.ubuntu.com/ubuntu noble/main amd64 liberror-perl all 0.17029-2 [25,6 kB]  
Obter:7 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 git-man all 1:2.43.0-1ubuntu7.2 [1.100 kB]  
Obter:8 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 git amd64 1:2.43.0-1ubuntu7.2 [3.679 kB]  
Obter:9 http://archive.ubuntu.com/ubuntu noble/universe amd64 ubuntu-fan all 0.12.16 [35,2 kB]  
Obter:5 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 docker.io amd64 27.5.1-0ubuntu3~24.04.2 [33,0 MB]  
86% [5 docker.io 20,7 MB/33,0 MB 63%]
```

Figura 2. Instalação do docker.io.

- `sudo systemctl start docker` – Esse comando iniciou o serviço do Docker imediatamente.
- `sudo systemctl enable docker` – Esse commando configurou o Docker para iniciar automaticamente toda vez que o Ubuntu for ligado.
- `sudo usermod -aG docker $USER` – Esse comando adicionou o usuário atual ao grupo de permissões chamado docker. Esses últimos três commands podem ser visualizados na figura 3.

```
daiane@daiane-VirtualBox: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
Configurando runc (1.2.5-0ubuntu1~24.04.1) ...  
Configurando liberror-perl (0.17029-2) ...  
Configurando bridge-utils (1.7.1-1ubuntu2) ...  
Configurando pigz (2.8-1) ...  
Configurando git-man (1:2.43.0-1ubuntu7.2) ...  
Configurando containerd (1.7.27-0ubuntu1~24.04.1) ...  
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service →  
  /usr/lib/systemd/system/containerd.service.  
Configurando ubuntu-fan (0.12.16) ...  
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service →  
  /usr/lib/systemd/system/ubuntu-fan.service.  
Configurando docker.io (27.5.1-0ubuntu3~24.04.2) ...  
info: Selecionando GID da faixa 100 a 999 ...  
info: Adicionando grupo 'docker' (GID 127) ...  
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /us  
r/lib/systemd/system/docker.service.  
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/li  
b/systemd/system/docker.socket.  
Configurando git (1:2.43.0-1ubuntu7.2) ...  
A processar 'triggers' para man-db (2.12.0-4build2) ...  
daiane@daiane-VirtualBox:~$ sudo systemctl start docker  
daiane@daiane-VirtualBox:~$ sudo systemctl enable docker  
daiane@daiane-VirtualBox:~$ sudo usermod -aG docker $USER  
daiane@daiane-VirtualBox:~$
```

Figura 3. Iniciando, configurando e adicionando usuário no docker.

- `sudo reboot` – Esse comando foi aplicado para executar as permissões corretamente. Para isso, ele reinicia o sistema.

Depois que o Sistema foi reiniciado, o passo a passo descrito nessa etapa 1 foi corretamente implementado. Então, foi possível criar um container com NGINX para ser usado a partir do docker dentro do VirtualBox. O passo a passo para a criação do container com NGINX está descrito na etapa 2.

2.2. Etapa 2 - Criando um container com o NGINX

1. No terminal do VirtualBox, executamos os comandos:

- `docker run -d -p 80:80 --name meu-nginx nginx` – Esse commando criou e iniciou um container do Docker com o servidor web NGINX rodando na porta 80, como mostra a figura 4. O nome personalizado do container criado é: `meu-nginx`.
- `docker ps` – Esse comando mostrou todos os containers do Docker que estavam em execução (ver figura 4).

```

daiane@daiane-VirtualBox: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
daiane@daiane-VirtualBox:~$ docker run -d -p 80:80 --name meu-nginx nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
dad67da3f26b: Pull complete
4eb3a9835b30: Pull complete
021db26e13de: Pull complete
397cc88dcd41: Pull complete
5f4a88bd8474: Pull complete
66467f827546: Pull complete
f05e87039331: Pull complete
Digest: sha256:dc53c8f25a10f9109190ed5b59bda2d707a3bde0e45857ce9e1efaa32ff9cbc1
Status: Downloaded newer image for nginx:latest
6c9e939eee308d73c2095e1b52998dae0f9385b54f31bde7b8c859a8a86e1b96
daiane@daiane-VirtualBox:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
PORTS         NAMES
6c9e939eee30   nginx    "/docker-entrypoint..." About a minute ago Up About
a minute      0.0.0.0:80->80/tcp, :::80->80/tcp meu-nginx
daiane@daiane-VirtualBox:~$

```

Figura 4. Criação do container com o Nginx e verificação dos containers em execução, respectivamente.

Para acessar o IP da MV através do navegador de qualquer dispositivo, nós mudamos a configuração de Rede na MV, conforme descrito na etapa 3.

2.3. Etapa 3 – Alterando a configuração de rede no VirtualBox

1. Desligamos a MV no VirtualBox.
2. No VirtualBox, selecionamos a Máquina Virtual Linux2025 e clicamos em Configurações.

3. Em seguida, selecionamos Rede e Adaptador1.
4. Na opção “Conectado a”, escolhemos Placa em modo bridge.
5. Em “Nome”, selecionamos a placa de rede Wi-Fi do notebook.
6. Clicamos em Ok e iniciamos a MV novamente.

Todos esses passos podem ser visualizados na figura 5.

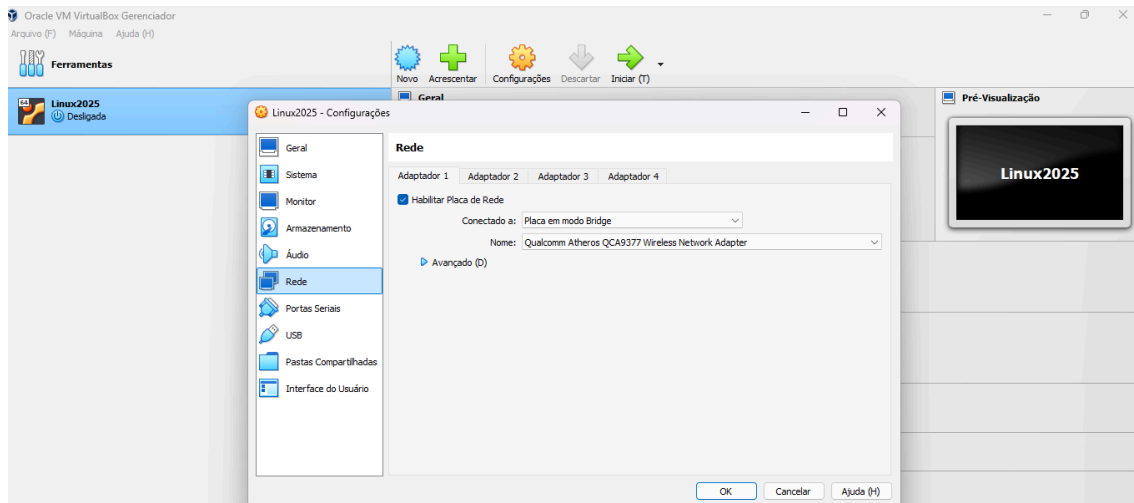


Figura 5. Configuração de rede no VirtualBox.

2.4. Etapa 4 – Descobrimo o IP da MV e acessando o NGINX pelo navegador de qualquer dispositivo.

1. Com a MV inicializada, executamos no terminal o comando: `ip a` - Esse comando nos apresentou o IP da MV. O IP da MV está destacado na figura 6.

```
daiane@daiane-VirtualBox:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:12:90:58 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.4/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s3
        valid_lft 7114sec preferred_lft 7114sec
    inet6 2804:29b8:5071:2896:bee1:47c9:9d93:1ef2/64 scope global temporary dynamic
        valid_lft 259198sec preferred_lft 86000sec
    inet6 2804:29b8:5071:2896:29da:5769:c03d:bf90/64 scope global dynamic mngtmpaddr noprefixroute
        valid_lft 259198sec preferred_lft 172798sec
    inet6 fe80::ed63:ee2:9783:e3ff/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:60:1c:26:69 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
    inet6 fe80::42:60ff:felc:2669/64 scope link
        valid_lft forever preferred_lft forever
5: veth917dfd9@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
    link/ether 76:33:67:e6:87:76 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::7433:67ff:fee6:8776/64 scope link
        valid_lft forever preferred_lft forever
```

Figura 6. Localizando o IP da MV.

A **enp0s3** é a interface de rede NAT padrão do VirtualBox. O inet, que está abaixo da interface **enp0s3**, mostra o endereço IPv4 da MV: 192.168.1.4.

2. Em seguida, digitamos `http:// 192.168.1.4`, no navegador do notebook e visualizamos a imagem apresentada na figura 7.

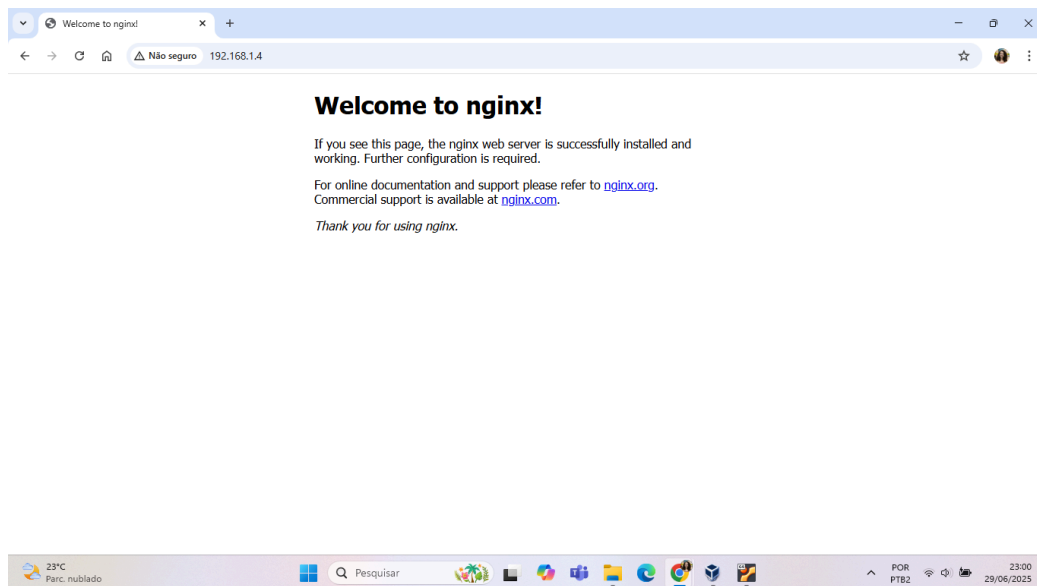


Figura 7. Página padrão no NGNIX.

A figura 7 mostra a página padrão do NGINX, indicando que o container com o NGINX foi instalado com sucesso na MV.

Em seguida, para personalizar o código da página `index.html`, tivemos que acessar o container do NGINX. Essa personalização está descrita na etapa 5.

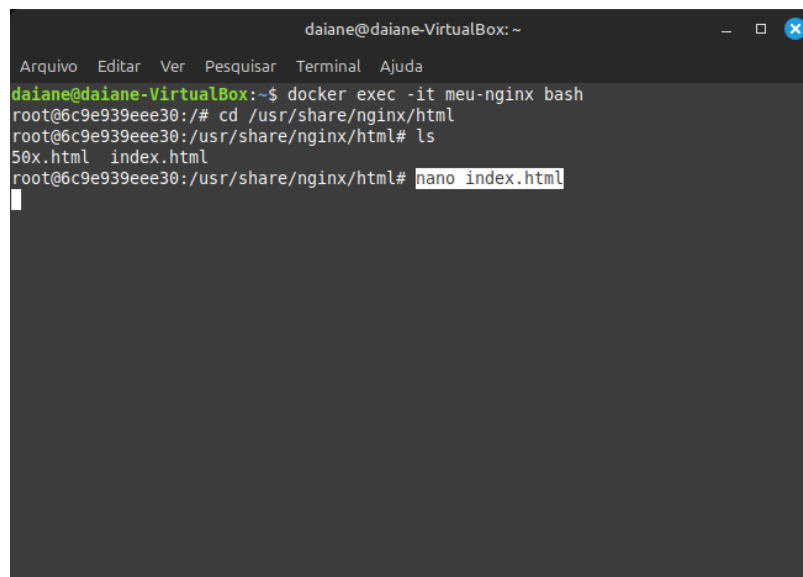
2.5. Etapa 5 – Personalizando a página `index.html` e criando novas pastas no NGINX.

1. No terminal da MV, executamos os comandos:

- `docker exec -it meu-nginx bash` – Esse comando abriu um terminal dentro do container chamado `meu-nginx`.
- `cd /usr/share/nginx/html` – Dentro do container, esse comando localizou a página onde estava o `index.html`.
- `ls` – Esse comando listou os arquivos da pasta.
- `nano index.html` – Com esse comando nós personalizamos o arquivo `index.html` com o editor `nano`.

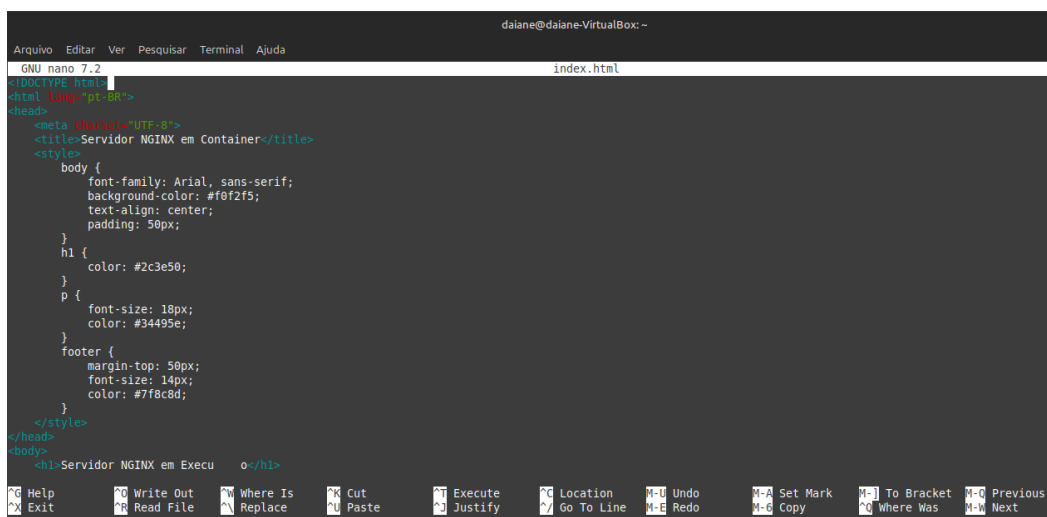
A execução de todos os comando, listados para essa etapa 5, estão ilustrados nas figuras 8 e 9.

Para salvar o arquivo `index.html` pressionamos `Ctrl + O` no teclado e em seguida pressionamos a tecla `Enter`. Para sair do referido arquivo pressionamos `Ctrl + X`.



```
daiane@daiane-VirtualBox: ~  
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda  
daiane@daiane-VirtualBox:~$ docker exec -it meu-nginx bash  
root@6c9e939eee30:/# cd /usr/share/nginx/html  
root@6c9e939eee30:/usr/share/nginx/html# ls  
50x.html  index.html  
root@6c9e939eee30:/usr/share/nginx/html# nano index.html
```

Figura 8. Acessando o arquivo index.html.



```
GNU nano 7.2 index.html  
<!DOCTYPE html>  
<html lang="pt-BR">  
<head>  
  <meta charset="UTF-8">  
  <title>Servidor NGINX em Container</title>  
  <style>  
    body {  
      font-family: Arial, sans-serif;  
      background-color: #f0f2f5;  
      text-align: center;  
      padding: 50px;  
    }  
    h1 {  
      color: #2c3e50;  
    }  
    p {  
      font-size: 18px;  
      color: #34495e;  
    }  
    footer {  
      margin-top: 50px;  
      font-size: 14px;  
      color: #7f8c8d;  
    }  
  </style>  
</head>  
<body>  
  <h1>Servidor NGINX em Execu    o</h1>
```

Figura 9. Personalizando no nano o arquivo index.html.

2. Para mostrar que é possível acessar o NGINX através do navegador de qualquer dispositivo, depois da atualização do arquivo index.html, nós acessamos o NGINX pelo navegador de um celular, a página atualizada pode ser verificada pela figura 10.

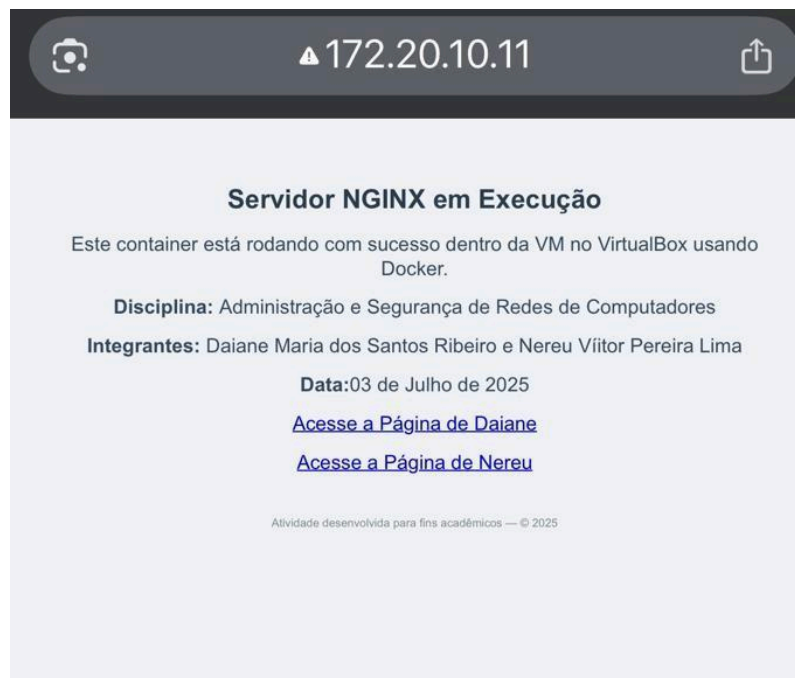


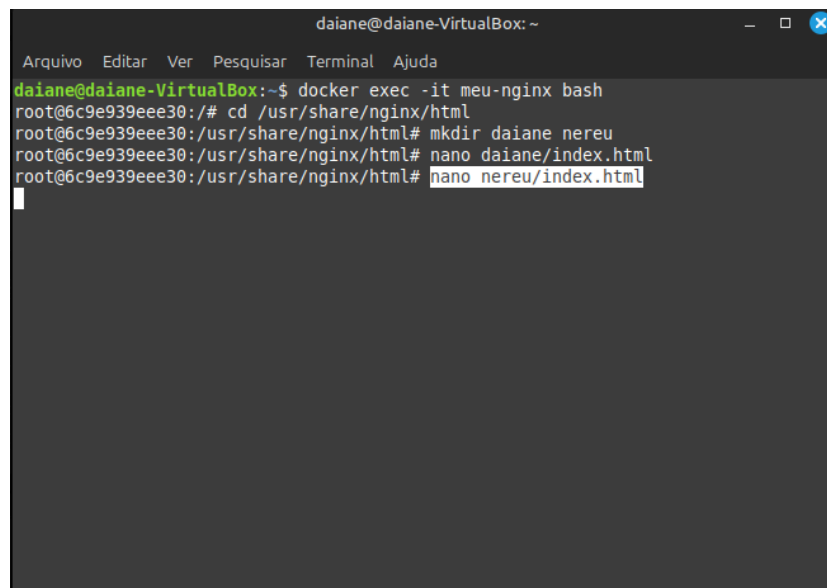
Figura 10. Arquivo index.html personalizado.

3. Em seguida, criamos as pastas daiane e nereu dentro do container, no mesmo diretório onde está o index.html principal, que é servido pelo NGINX. Em cada uma das pastas (daiane, nereu) foram criados arquivos index.html, que podem ser acessados através do index.html principal. Para criarmos essas duas pastas, com os arquivos index.html, executamos no terminal da MV os comandos:

- `docker exec -it meu-nginx bash` - Esse comando abriu um terminal dentro do container chamado meu-nginx.
- `cd /usr/share/nginx/html` - Esse comando acessou o diretório HTML do Nginx.
- `mkdir daiane nereu` - Esse comando criou as pastas daiane e nereu.
- `nano daiane/index.html` - Por meio desse comando criamos e personalizamos a pasta e o arquivo daiane/index.html. Em seguida, salvamos o arquivo com `Ctrl + O` e `Enter` e saímos do arquivo com `Ctrl + X`.
- `nano nereu/index.html` - Com esse comando criamos e personalizamos a pasta e o arquivo nereu/index.html. Em seguida, salvamos o arquivo com `Ctrl + O` e `Enter` e saímos do arquivo com `Ctrl + X`.

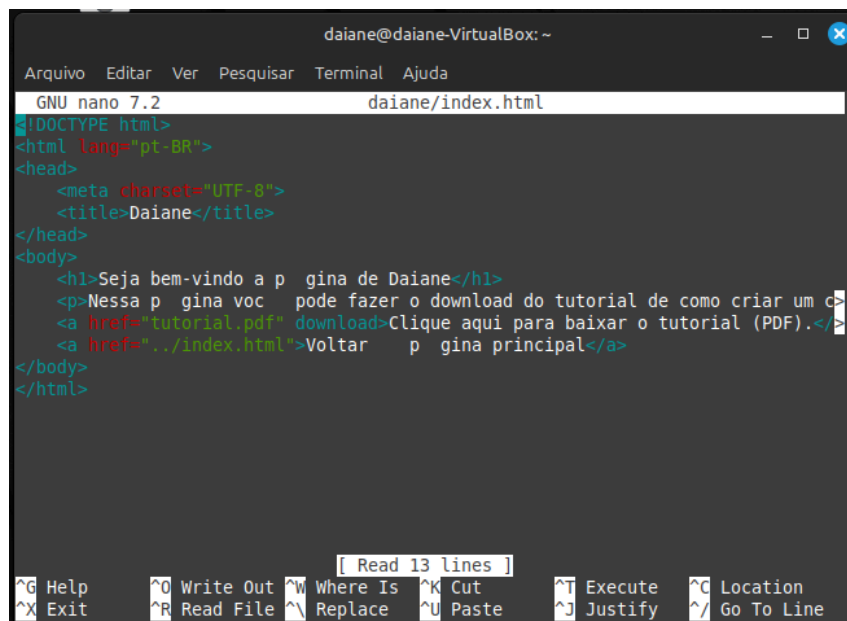
A execução de todos os comandos descritos nesse passo está ilustrada nas figuras 11, 12 e 13.

Depois, atualizamos o index.html principal com os links do index.html da pasta daiane e nereu.



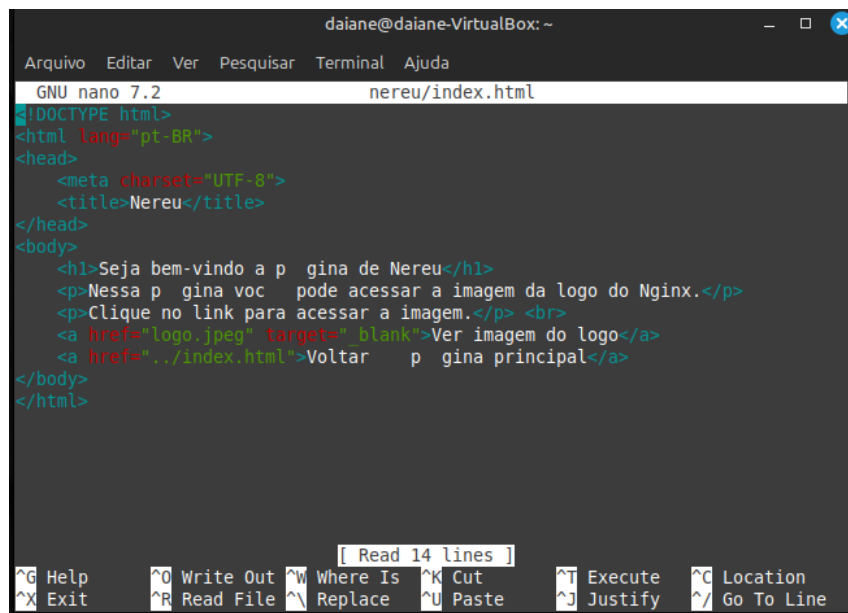
```
daiane@daiane-VirtualBox: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
daiane@daiane-VirtualBox:~$ docker exec -it meu-nginx bash  
root@6c9e939eee30:/# cd /usr/share/nginx/html  
root@6c9e939eee30:/usr/share/nginx/html# mkdir daiane nereu  
root@6c9e939eee30:/usr/share/nginx/html# nano daiane/index.html  
root@6c9e939eee30:/usr/share/nginx/html# nano nereu/index.html
```

Figura 11. Criando a pasta daiane e nereu e o arquivo index em cada pasta.



```
daiane@daiane-VirtualBox: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
GNU nano 7.2 daiane/index.html  
!DOCTYPE html>  
<html lang="pt-BR">  
<head>  
  <meta charset="UTF-8">  
  <title>Daiane</title>  
</head>  
<body>  
  <h1>Seja bem-vindo a p gina de Daiane</h1>  
  <p>Nessa p gina voc pode fazer o download do tutorial de como criar um c</p>  
  <a href="tutorial.pdf" download>download</a>Clique aqui para baixar o tutorial (PDF).</a>  
  <a href=" ../index.html">Voltar p gina principal</a>  
</body>  
</html>  
[ Read 13 lines ]  
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location  
^X Exit ^R Read File ^_ Replace ^U Paste ^J Justify ^_ Go To Line
```

Figura 12. Personalizando o index.html na pasta daiane.



```
daiane@daiane-VirtualBox: ~
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
GNU nano 7.2      nereu/index.html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <title>Nereu</title>
</head>
<body>
  <h1>Seja bem-vindo a página de Nereu</h1>
  <p>Nessa página você pode acessar a imagem da logo do Nginx.</p>
  <p>Clique no link para acessar a imagem.</p> <br>
  <a href="logo.jpeg" target="_blank">Ver imagem do logo</a>
  <a href=" ../index.html">Voltar à página principal</a>
</body>
</html>
[ Read 14 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

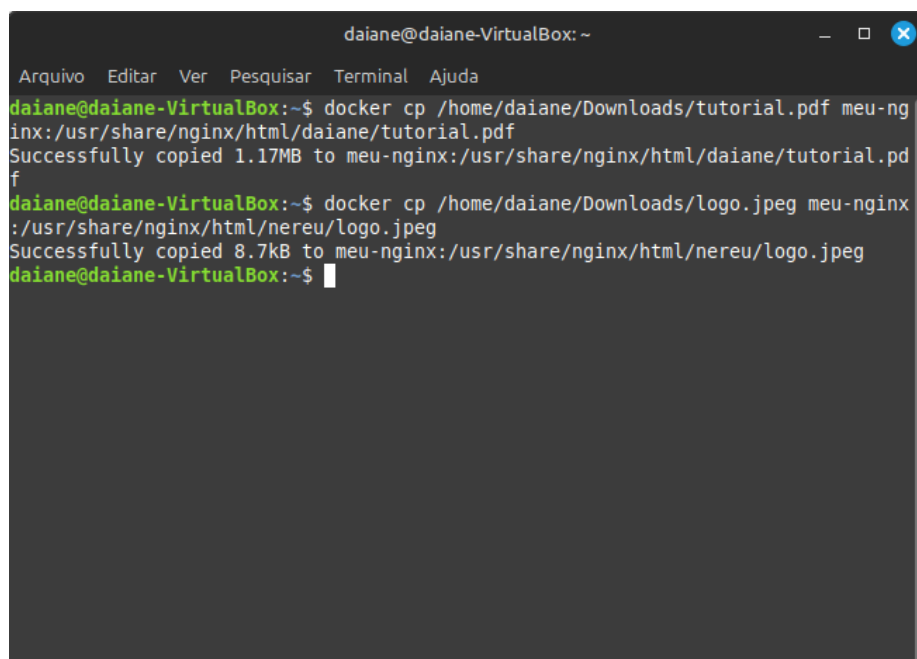
Figura 13. Personalizando o index.html na pasta daiane.

Pelas figuras 12 e 13 é possível perceber que nós copiamos arquivos externos para dentro do container com o NGINX, pois no arquivo index.html da pasta daiane, podemos acessar esse tutorial em pdf e no arquivo index.html da pasta nereu, é possível acessar a imagem da logo do NGINX. Iremos explicar como fizemos isso no passo 4.

4. O arquivo tutorial.pdf e logo.Jpeg foram salvos na MV, na pasta download. Então, executamos no terminal o comando:

- `docker cp /home/daiane/Downloads/tutorial.pdf meu-nginx:/usr/share/nginx/html/daiane/tutorial.pdf` – Esse comando foi usado fora do container para mover o arquivo tutorial.pdf para a pasta /usr/share/nginx/html/daiane/ dentro do container meu-nginx.
- `docker cp /home/daiane/Downloads/logo.jpeg meu-nginx:/usr/share/nginx/html/nereu/logo.jpeg` – Esse comando foi usado fora do container para mover o arquivo logo.jpeg para a pasta /usr/share/nginx/html/nereu/ dentro do container meu-nginx.

A figura 14 mostra a implementação dos comandos acima, funcionando corretamente.



```
daiane@daiane-VirtualBox: ~  
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda  
daiane@daiane-VirtualBox:~$ docker cp /home/daiane/Downloads/tutorial.pdf meu-nginx:/usr/share/nginx/html/daiane/tutorial.pdf  
Successfully copied 1.17MB to meu-nginx:/usr/share/nginx/html/daiane/tutorial.pdf  
daiane@daiane-VirtualBox:~$ docker cp /home/daiane/Downloads/logo.jpeg meu-nginx:/usr/share/nginx/html/nereu/logo.jpeg  
Successfully copied 8.7kB to meu-nginx:/usr/share/nginx/html/nereu/logo.jpeg  
daiane@daiane-VirtualBox:~$
```

Figura 14. Copiando o arquivo tutorial.pdf e logo.jpeg para dentro do container.

Assim, foi possível acessar o arquivo tutorial.pdf pela página index.html da pasta daiane e acessar o arquivo logo.jpeg pela página index.html da pasta nereu, como mostram as figuras 15 e 16, respectivamente.



Figura 15. Página index.html da pasta daiane.

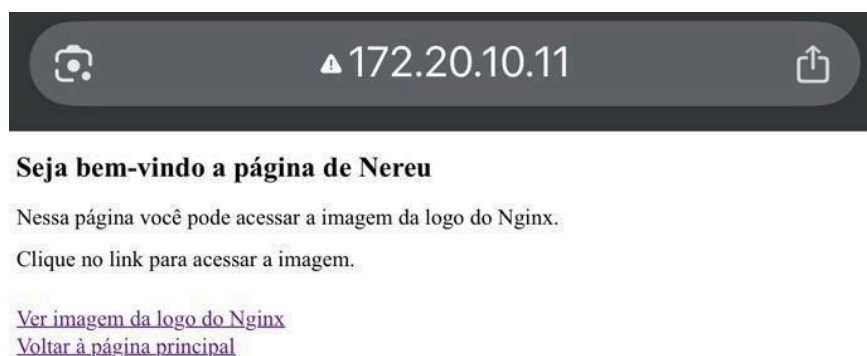


Figura 16. Página index.html da pasta nereu.

Além disso, o projeto desenvolvido foi colocado no GitHub, que é uma plataforma gratuita onde é possível publicar arquivos. A descrição de como subir o projeto nessa nuvem está apresentada na etapa 6.

2.6. Etapa 6 – Compartilhando o projeto no GitHub.

1. Acessamos o site do GitHub: <https://github.com/> e logamos no Sistema.
2. Criamos um novo repositório (ver figura 17):
 - Nome: servidor-nginx-docker
 - Deixamos como público
 - Clicamos em Criar repositório



A imagem mostra a interface de criação de um novo repositório no GitHub. No topo, há uma barra de navegação com o endereço "github.com/new". O título principal é "Criar um novo repositório". Abaixo dele, há uma explicação sobre repositórios e um link para "Importe um repositório".

Os campos obrigatórios são marcados com um asterisco (*).

Proprietário * / Nome do repositório *

O proprietário é "daianesr35" e o nome do repositório é "servidor-nginx-docker". Há uma mensagem de confirmação: "servidor-nginx-docker está disponível".

Ótimos nomes de repositórios são curtos e fáceis de lembrar. Precisa de inspiração? Que tal [spork animado](#) ?

Descrição (opcional)

Existem duas opções de visibilidade: "Público" (selecionada) e "Privado".

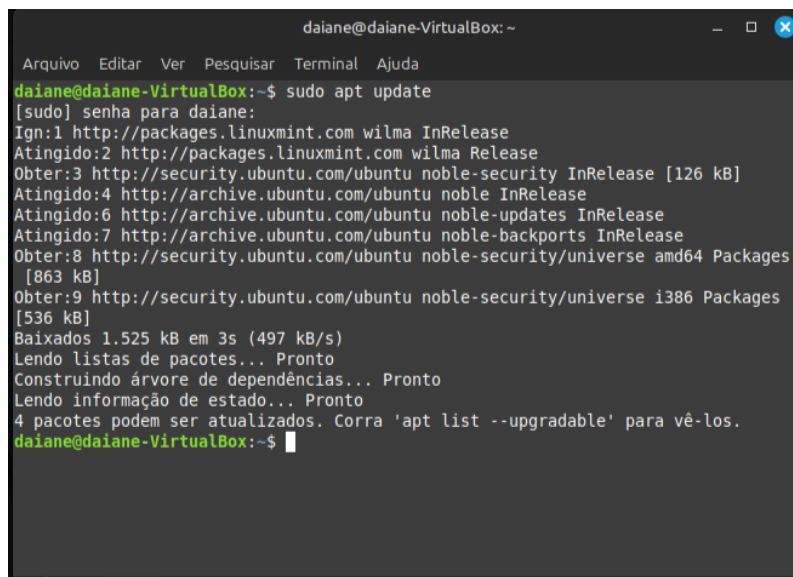
Inicialize este repositório com:

☒ Adicionar um arquivo README

Abaixo disso, há um link para "Saiba mais sobre READMEs" e uma opção para "Adicionar .gitignore".

Figura 17. Criando um novo repositório no GitHub.

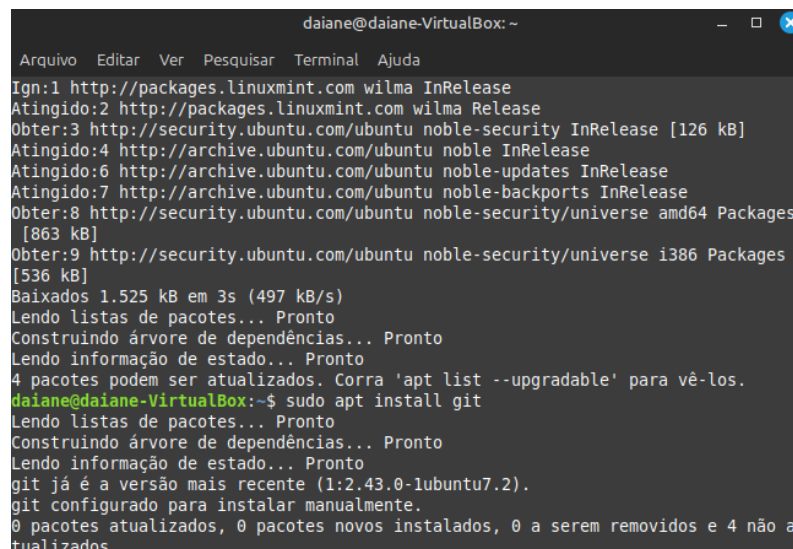
3. Na MV, instalamos o GitHub. Para isso, executamos os seguintes comandos no terminal da MV:
 - `sudo apt update` – Esse comando atualizou a lista de pacotes disponíveis nos repositórios do Ubuntu, como mostra a figura 18.



```
daiane@daiane-VirtualBox: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
daiane@daiane-VirtualBox:~$ sudo apt update
[sudo] senha para daiane:
Ign:1 http://packages.linuxmint.com wilma InRelease
Atingido:2 http://packages.linuxmint.com wilma Release
Obter:3 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Atingido:4 http://archive.ubuntu.com/ubuntu noble InRelease
Atingido:6 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Atingido:7 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Obter:8 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages
[863 kB]
Obter:9 http://security.ubuntu.com/ubuntu noble-security/universe i386 Packages
[536 kB]
Baixados 1.525 kB em 3s (497 kB/s)
Lendo listas de pacotes... Pronto
Construindo árvore de dependências... Pronto
Lendo informação de estado... Pronto
4 pacotes podem ser atualizados. Corra 'apt list --upgradable' para vê-los.
daiane@daiane-VirtualBox:~$
```

Figura 18. Atualização de lista de pacotes da MV.

- `sudo apt install git` – Esse comando instalou o Git, que é o sistema de controle de versão usado para trabalhar com repositórios no GitHub, como ilustra a figura 19.



```
daiane@daiane-VirtualBox: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
Ign:1 http://packages.linuxmint.com wilma InRelease
Atingido:2 http://packages.linuxmint.com wilma Release
Obter:3 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Atingido:4 http://archive.ubuntu.com/ubuntu noble InRelease
Atingido:6 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Atingido:7 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Obter:8 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages
[863 kB]
Obter:9 http://security.ubuntu.com/ubuntu noble-security/universe i386 Packages
[536 kB]
Baixados 1.525 kB em 3s (497 kB/s)
Lendo listas de pacotes... Pronto
Construindo árvore de dependências... Pronto
Lendo informação de estado... Pronto
4 pacotes podem ser atualizados. Corra 'apt list --upgradable' para vê-los.
daiane@daiane-VirtualBox:~$ sudo apt install git
Lendo listas de pacotes... Pronto
Construindo árvore de dependências... Pronto
Lendo informação de estado... Pronto
git já é a versão mais recente (1:2.43.0-1ubuntu7.2).
git configurado para instalar manualmente.
0 pacotes atualizados, 0 pacotes novos instalados, 0 a serem removidos e 4 não a
tualizados.
```

Figura 19. Instalação do Git na MV.

4. Configuramos o Git, por meio dos comandos:

- `git config --global user.name "daiane"` – Esse comando definiu o nome de usuário que será associado a todos os commits feitos com o Git na MV usada.
- `git config --global user.email "daiane.ribeiro@ifsertao-pe.edu.br"` – Esse comando definiu o e-mail do usuário que será associado aos commits.

5. Copiamos o index.html do container com NGINX para a pasta pessoal na MV. No terminal da MV, executamos:

- `docker cp meu-nginx:/usr/share/nginx/html/index.html ~/index.html` – Esse comando copiou o arquivo index.html do container para a pasta pessoal da MV.

6. Criamos uma nova pasta para o repositório dentro da MV, através do comando:

- `mkdir ~/projeto-nginx` – Esse comando criou uma nova pasta chamada projeto-nginx dentro da pasta pessoal na MV.
- `cd ~/projeto-nginx` – Esse comando permitiu entrar na pasta projeto-nginx para começar a trabalhar dentro dela (criar arquivos, iniciar um repositório Git, etc).

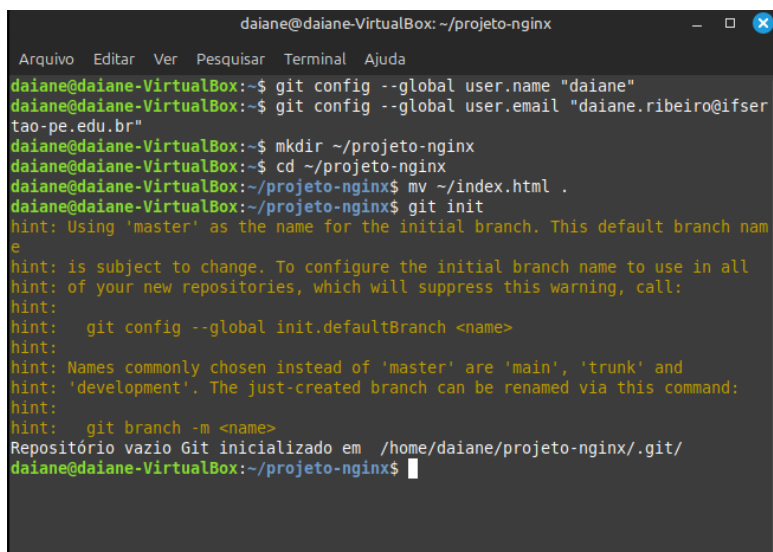
7. Movemos o index.html para dentro da pasta criada no passo anterior:

- `mv ~/index.html .` – Esse comando moveu o arquivo index.html da pasta pessoal para o diretório projeto-nginx.

8. Iniciamos o GitHub na pasta projeto-nginx, por meio do comando abaixo:

- `git init`

A figura 20 ilustra a execução dos passos 4, 6, 7 e 8 dessa etapa.



```
daiane@daiane-VirtualBox: ~/projeto-nginx
Arquivo  Editar  Ver      Pesquisar  Terminal  Ajuda
daiane@daiane-VirtualBox:~$ git config --global user.name "daiane"
daiane@daiane-VirtualBox:~$ git config --global user.email "daiane.ribeiro@ifser
tao-pe.edu.br"
daiane@daiane-VirtualBox:~$ mkdir ~/projeto-nginx
daiane@daiane-VirtualBox:~$ cd ~/projeto-nginx
daiane@daiane-VirtualBox:~/projeto-nginx$ mv ~/index.html .
daiane@daiane-VirtualBox:~/projeto-nginx$ git init
hint: Using 'master' as the name for the initial branch. This default branch nam
e
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Repositório vazio Git inicializado em /home/daiane/projeto-nginx/.git/
daiane@daiane-VirtualBox:~/projeto-nginx$
```

Figura 20. Configuração do Git na MV.

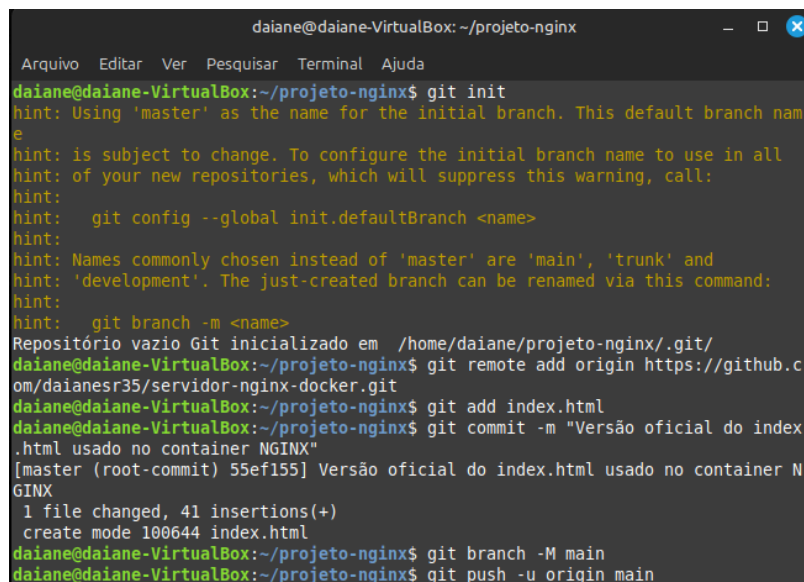
9. Conectamos o repositório local ao repositório remoto no GitHub, usando o comando abaixo:

- `git remote add origin https://github.com/daianesr35/servidor-nginx-docker.git`

10. Criamos o commit e enviamos para o GitHub:

- `git add index.html` - Esse comando adicionou o arquivo `index.html` à área de preparação do Git, indicando que ele está pronto para ser incluído no próximo commit.
- `git commit -m "Versão oficial do index.html usado no container NGINX"` - Esse comando registrou oficialmente a versão do `index.html` no repositório local, criando um ponto no histórico do projeto com uma mensagem explicativa.
- `git branch -M main` - Esse comando definiu o nome do branch principal como `main`, renomeando-o se for necessário.
- `git push -u origin main` - Esse comando enviou o commit feito localmente para o repositório remoto no GitHub, no branch `main`.

A execução dos comandos dos passos 9 e 10 está ilustrada na figura 21.



```

daiane@daiane-VirtualBox: ~/projeto-nginx
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
daiane@daiane-VirtualBox:~/projeto-nginx$ git init
hint: Using 'master' as the name for the initial branch. This default branch nam
e
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Repositório vazio Git inicializado em /home/daiane/projeto-nginx/.git/
daiane@daiane-VirtualBox:~/projeto-nginx$ git remote add origin https://github.c
om/daianesr35/servidor-nginx-docker.git
daiane@daiane-VirtualBox:~/projeto-nginx$ git add index.html
daiane@daiane-VirtualBox:~/projeto-nginx$ git commit -m "Versão oficial do index
.html usado no container NGINX"
[master (root-commit) 55ef155] Versão oficial do index.html usado no container N
GINX
 1 file changed, 41 insertions(+)
 create mode 100644 index.html
daiane@daiane-VirtualBox:~/projeto-nginx$ git branch -M main
daiane@daiane-VirtualBox:~/projeto-nginx$ git push -u origin main

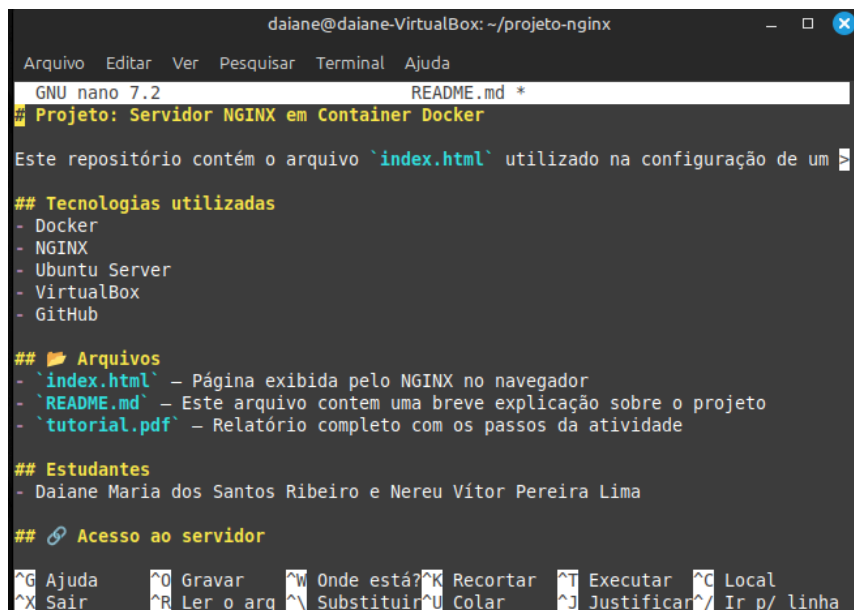
```

Figura 21. Conexão do repositório local com o repositório remoto.

11. Criamos o arquivo `README.md` com uma breve explicação do projeto, como mostra a figura 22. Na MV executamos:

- `nano README.md`

Depois pressionamos `Ctrl + O` e em seguida `Enter` para salvar o arquivo criado e pressionamos `Ctrl + X` para sair do arquivo.



```
daiane@daiane-VirtualBox: ~/projeto-nginx
GNU nano 7.2 README.md *
Projeto: Servidor NGINX em Container Docker

Este repositório contém o arquivo `index.html` utilizado na configuração de um >

## Tecnologias utilizadas
- Docker
- NGINX
- Ubuntu Server
- VirtualBox
- GitHub

## 📁 Arquivos
- `index.html` - Página exibida pelo NGINX no navegador
- `README.md` - Este arquivo contém uma breve explicação sobre o projeto
- `tutorial.pdf` - Relatório completo com os passos da atividade

## Estudantes
- Daiane Maria dos Santos Ribeiro e Nereu Vítor Pereira Lima

## 🔗 Acesso ao servidor

^G Ajuda  ^O Gravar  ^W Onde está? ^K Recortar  ^T Executar  ^C Local
^X Sair    ^R Ler o arq ^\ Substituir ^U Colar    ^J Justificar ^_ Ir p/ linha
```

Figura 22. Editando o arquivo README.md.

12. Copiamos o arquivo tutorial.pdf para a pasta do projeto. Como o arquivo tutorial.pdf estava na pasta pessoal, usamos o comando abaixo para copiar o arquivo para a pasta projeto-nginx:

- `cp ~/tutorial.pdf ~/projeto-nginx/`

13. Adicionamos e enviamos os arquivos README.md e tutorial.pdf para o GitHub, através dos comando abaixo:

- `git add README.md tutorial.pdf`
- `git commit -m "Adicionando README.md e tutorial.pdf ao repositório"`
- `git push origin main`

14. Verificamos no GitHub e notamos a presença dos arquivos index.html, README.md e tutorial.pdf, conforme mostra a figura 23.

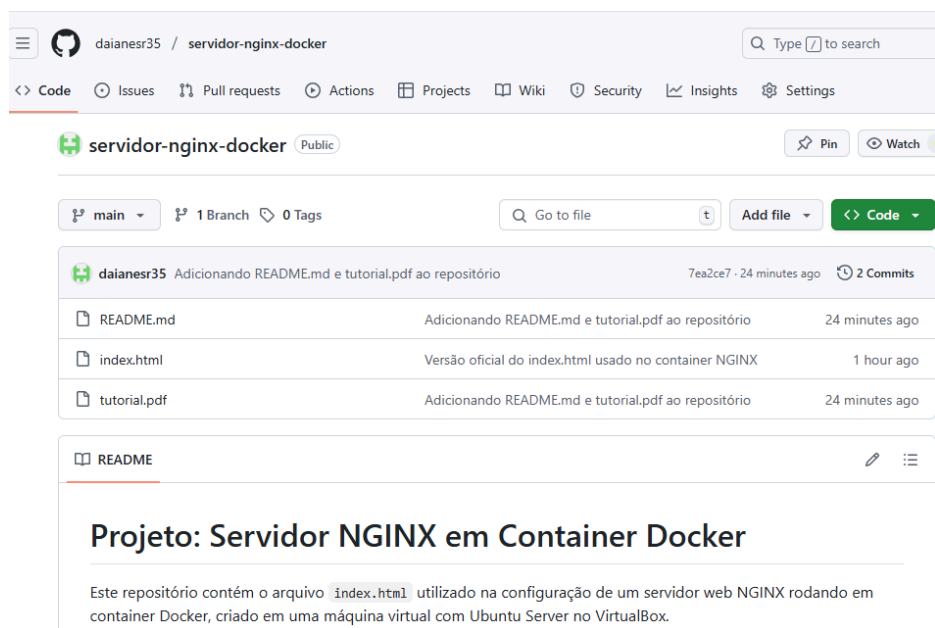


Figura 23. Arquivos sincronizados no GitHub.

Para finalizar esse tutorial nós iremos mostrar como compartilhar o container com o NGINX no DockerHub. Uma vez que não é possível subir uma *ISO* do container NGINX para o GitHub, por dois motivos:

1. **Containers Docker não geram arquivos ISO** (ISO é uma imagem de disco, usada para instalar sistemas operacionais).
2. **O GitHub não é feito para armazenar imagens grandes de disco ou containers prontos**, e há limite de tamanho por arquivo (100 MB via interface web, e até 2 GB via Git LFS, com restrições).

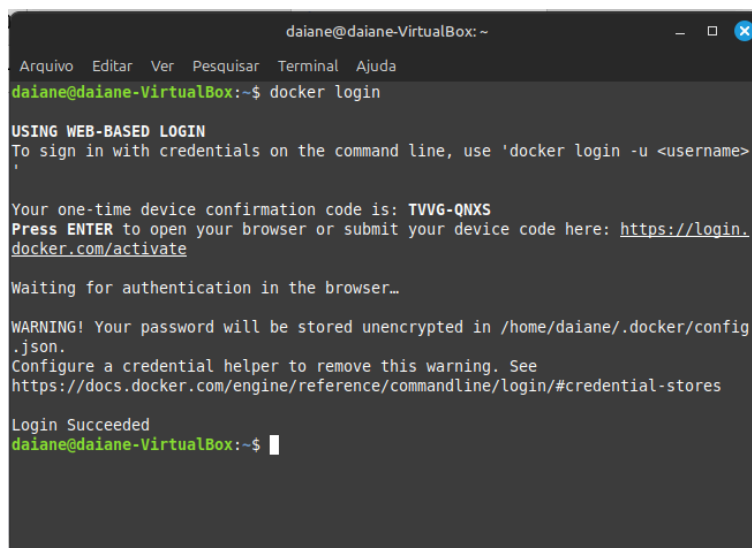
2.6. Etapa 6 – Compartilhando o projeto no DockerHub

1. Acessamos o site do dockerhub: <https://hub.docker.com> e entramos no sistema usando a conta do GitHub.

2. Fizemos login no Docker pela MV, executando o seguinte comando no terminal:

- `docker login`

A figura 24 mostra o comando sendo executando e o login efetuado com sucesso no dockerhub.

A terminal window titled 'daiane@daiane-VirtualBox: ~' with a menu bar (Arquivo, Editar, Ver, Pesquisar, Terminal, Ajuda). The user enters 'docker login'. The terminal shows the Docker login process: 'USING WEB-BASED LOGIN', instructions to sign in, a one-time device confirmation code 'TVVG-QNXS', a link to 'https://login.docker.com/activate', and a warning about password storage. It then says 'Waiting for authentication in the browser...' and 'Login Succeeded'.

```
daiane@daiane-VirtualBox:~$ docker login

USING WEB-BASED LOGIN
To sign in with credentials on the command line, use 'docker login -u <username>'

Your one-time device confirmation code is: TVVG-QNXS
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate

Waiting for authentication in the browser...

WARNING! Your password will be stored unencrypted in /home/daiane/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
daiane@daiane-VirtualBox:~$
```

Figura 23. Login no dockerhub na MV.

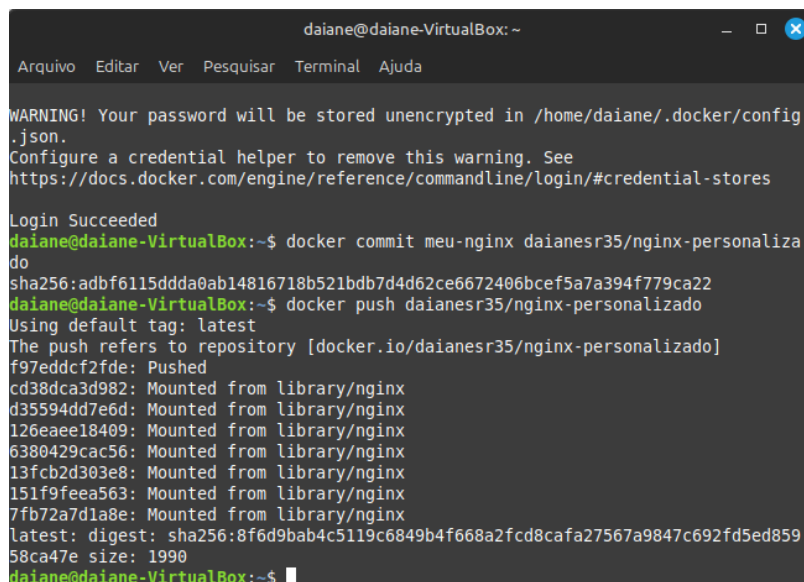
3. Criamos uma imagem a partir do container com NGINC na MV, usando no terminal o comando:

- `docker commit meu-nginx daianesr35/nginx-personalizado`

4. Enviamos a imagem para o dockerhub, através do comando:

- `docker push daianesr35/nginx-personalizado`

A figura 24 ilustra a criação da imagem a partir do container da MV e o envio do container com NGINX para o dockerhub.

A terminal window titled 'daiane@daiane-VirtualBox: ~' with a menu bar (Arquivo, Editar, Ver, Pesquisar, Terminal, Ajuda). It shows the output of the 'docker commit' command, including a long sha256 hash. Then the user enters 'docker push daianesr35/nginx-personalizado'. The terminal shows the push process: 'Using default tag: latest', 'The push refers to repository [docker.io/daianesr35/nginx-personalizado]', and a list of layers being pushed from the library/nginx image, including the latest digest and size.

```
daiane@daiane-VirtualBox:~$ docker commit meu-nginx daianesr35/nginx-personalizado
sha256:adb6f115ddda0ab14816718b521bdb7d4d62ce6672406bcef5a7a394f779ca22
daiane@daiane-VirtualBox:~$ docker push daianesr35/nginx-personalizado
Using default tag: latest
The push refers to repository [docker.io/daianesr35/nginx-personalizado]
f97eddcf2fde: Pushed
cd38dca3d982: Mounted from library/nginx
d35594dd7e6d: Mounted from library/nginx
126eae18409: Mounted from library/nginx
6380429cac56: Mounted from library/nginx
13fcb2d303e8: Mounted from library/nginx
151f9feea563: Mounted from library/nginx
7fb72a7d1a8e: Mounted from library/nginx
latest: digest: sha256:8f6d9bab4c5119c6849b4f668a2fcd8cfa27567a9847c692fd5ed85958ca47e size: 1990
daiane@daiane-VirtualBox:~$
```

Figura 24. Criação e envio da imagem do container para o dockerhub.

Podemos ver através da figura 25 que a imagem do container com NGINX foi compartilhada na plataforma dockerhub.

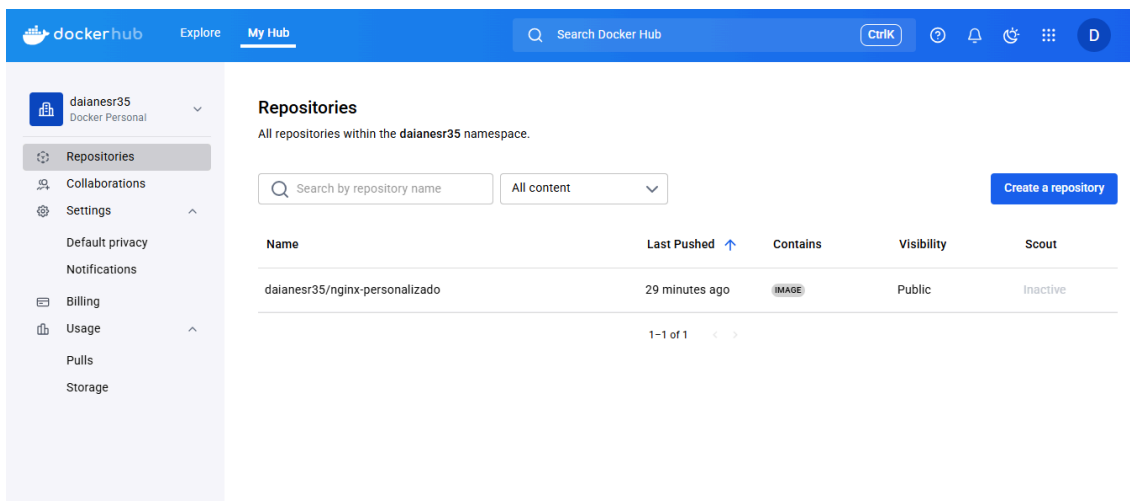


Figura 25. Imagem do container disponível no dockerhub.

O tamanho do container com NGINX que foi criado nesse projeto foi 85,3MB, como é possível observar na figura 26.

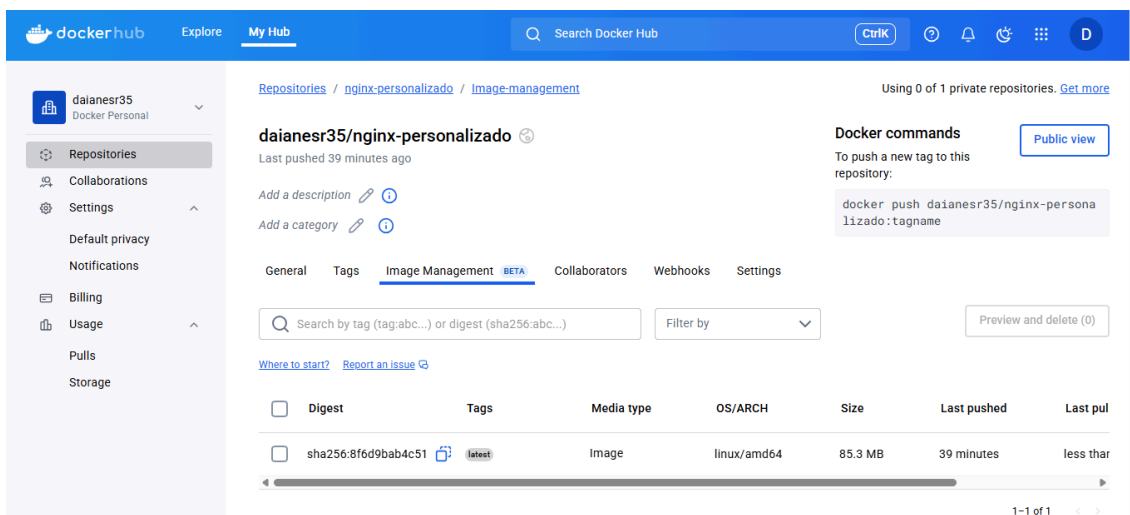


Figura 26. Imagem do container disponível no dockerhub.

5. Puxamos e executamos o container com NGINX usando outra MV. Para isso usamos os comandos:

- `Sudo docker pull daianesr35/nginx-personalizado`
- `Sudo docker run -d -p 80:80 --name teste-nginx daianesr35/nginx-personalizado`

A figura 27 ilustra o container com NGINX sendo puxado do dockerhub pela MV de Nereu .

```
nereu@nereu-VirtualBox: ~  
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda  
nereu@nereu-VirtualBox:~$ sudo docker pull daianesr35/nginx-personalizado:latest  
latest: Pulling from daianesr35/nginx-personalizado  
dad67da3f26b: Pull complete  
4eb3a9835b30: Pull complete  
021db26e13de: Pull complete  
397cc88dcd41: Pull complete  
5f4a88bd8474: Pull complete  
66467f827546: Pull complete  
f05e87039331: Pull complete  
6dc1bce6278c: Pull complete  
Digest: sha256:8f6d9bab4c5119c6849b4f668a2fcd8cafa27567a9847c692fd5ed85958ca47e  
Status: Downloaded newer image for daianesr35/nginx-personalizado:latest  
docker.io/daianesr35/nginx-personalizado:latest  
nereu@nereu-VirtualBox:~$
```

Figura 27. Puxando o container com NGINX usando a MV de Nereu.

A figura 28 mostra o container com NGINX sendo executando pela MV de Nereu.

```
nereu@nereu-VirtualBox: ~  
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda  
docker: Error response from daemon: pull access denied for daianesr35-personalizado, repository does not exist or may require 'docker login': denied: requested access to the resource is denied.  
See 'docker run --help'.  
nereu@nereu-VirtualBox:~$ sudo docker run -d -p 8080:80 --name meu-nginx/daianesr35-personalizado  
[sudo] senha para nereu:  
"docker run" requires at least 1 argument.  
See 'docker run --help'.  
  
Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]  
  
Create and run a new container from an image  
nereu@nereu-VirtualBox:~$ sudo docker run -d -p 8080:80 --name meu-nginx daianesr35/nginx-personalizado  
44ccdabf59a0e78068594491b240dc50d0622b60b8f13f53073fc3ba8b8471d3  
nereu@nereu-VirtualBox:~$ sudo docker ps  
CONTAINER ID    IMAGE                                COMMAND                  CREATED  
STATUS          PORTS                NAME  
S              44ccdabf59a0    daianesr35/nginx-personalizado  "/docker-entrypoint..."  About a minute ago  
Up About a minute    0.0.0.0:8080->80/tcp, [::]:8080->80/tcp  meu-nginx  
nereu@nereu-VirtualBox:~$
```

Figura 28. Executando o container com NGINX usando a MV de Nereu.

6. Por fim, acessamos o navegador da MV de Nereu e digitamos: <http://localhost:8080> e verificamos a página index.html principal do container com NGINX, ver figura 29. Então, com o projeto compartilhado no dockerhub ele pode ser acessado por qualquer pessoa que tenha interesse em usar o container com NGINX que criamos nesse projeto.

O link do repositório no dockerhub é: <https://hub.docker.com/repositories/daianesr35>.



Figura 29. Arquivo index.html do container com NGINX sendo acessado pelo navegador na MV de Nereu.

Considerações Finais

A realização desta atividade possibilitou a compreensão prática e detalhada do processo de implantação de um servidor web com NGINX utilizando contêineres Docker em uma máquina virtual no VirtualBox, com sistema operacional Linux. Durante o desenvolvimento do tutorial, foi possível observar as vantagens da containerização, como leveza, portabilidade, rapidez na execução e facilidade de replicação do ambiente.

A criação do container com NGINX permitiu não apenas simular o funcionamento de um servidor web real, como também possibilitou a personalização do conteúdo servido, com a modificação do arquivo index.html e a adição de novos diretórios e arquivos acessíveis pelo navegador, inclusive a partir de dispositivos externos à máquina virtual.

Outro ponto relevante foi a disponibilização do projeto no GitHub, o que garante a rastreabilidade das versões e facilita a colaboração, compartilhamento e reuso do ambiente por outros usuários. A integração entre Docker e GitHub reforça o uso de boas práticas no desenvolvimento de sistemas em ambientes modernos, como a computação em nuvem.

Portanto, a atividade proporcionou uma vivência prática importante sobre administração e segurança de redes, contribuindo para a formação técnica e crítica quanto à construção, publicação e manutenção de serviços em ambientes virtuais e distribuídos.

Referências

W3C – WORLD WIDE WEB CONSORTIUM. (2023) “HTTP – Hypertext Transfer Protocol Overview”, <https://www.w3.org/Protocols/>, May.

SUSNJARA, S.; SMALLEY, I. (2025). “O que são contêineres?”, <https://www.ibm.com/br-pt/think/topics/containers>, July

SUSNJARA, S.; SMALLEY, I. (2025). “O que é Docker?”, <https://www.ibm.com/br-pt/think/topics/docker>. July.

GITHUB (2025). “Sobre o GitHub e o Git”, <https://docs.github.com/pt/get-started/start-your-journey/about-github-and-git>, July.